DISEÑO DE CENTRO DE CONTROL VERSÁTIL PARA LA MONITORIZACIÓN Y CONTROL DE VEHÍCULOS AUTÓNOMOS

JUAN ANTONIO BONACHE SECO

MÁSTER EN INVESTIGACIÓN EN INFORMÁTICA, FACULTAD DE INFORMÁTICA,
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Ingeniería Informática para la Industria

23 de Junio de 2014

Directores: José Antonio López Orozco
Eva Besada Portas

Autorización de Difusión

JUAN ANTONIO BONACHE SECO

23 de Junio de 2014

El/la abajo firmante, matriculado/a en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "**DISEÑO DE CENTRO DE CONTROL VESÁTIL PARA LA MONITORIZACIÓN Y CONTROL DE VEHÍCULOS AUTÓNOMOS**", realizado durante el curso académico 2013-2014 bajo la dirección de los doctores **José Antonio López Orozco** y **Eva Besada Portas** en el Departamento de Arquitectura de Computadores y Automatismos, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Firmado: **Juan Antonio Bonache Seco**

AGRADECIMIENTOS

Gracias a los miembros del Departamento de Arquitectura de Computadores y Automatismos de la Universidad Complutense de Madrid por su ayuda y consejo, en especial a Eva y Jose Antonio, sin cuya experiencia y supervisión este proyecto no habría sido posible. También quisiera agradecer a David su consejo y paciencia durante la configuración del software de Simulación.

RESUMEN

El objetivo de este Trabajo de Fin de Máster es diseñar un prototipo de Centro de Control Versátil que permita la inclusión de vehículos autónomos de naturaleza heterogénea y la reconfiguración de elementos gráficos de monitorización y control de forma que se mejore la ergonomía del operador.

Un centro de control debe realizar la monitorización de datos de telemetría (p. ej. velocidad, longitud, latitud, orientación) además de proporcionar los elementos de control necesarios para dirigir los vehículos monitorizados. Este tipo de software suele estar estrechamente ligado al vehículo al que monitoriza debido a la heterogeneidad de sus controladores y primitivas de envío de datos de telemetría.

Por tanto, el reto al que nos enfrentamos en este proyecto, es lograr una estructura lo suficientemente flexible como para lograr monitorizar y controlar cualquier número de vehículos heterogéneos independientemente del hardware y software que los formen.

Esto lo lograremos en el Centro de Control Versátil mediante un software que extraerá de un documento XML los parámetros necesarios para su configuración, un interfaz gráfico de usuario reconfigurable y una estructura de comunicaciones sencilla y versátil.

SUMMARY

The goal of this Essay is to design a prototype of (GCS) Ground Control Station that allow us to include multiple heterogeneous unmanned vehicles and the reconfiguration of monitorization and control graphic elements in order to improve the operator's ergonomics.

A GCS must monitorize the telemetry data (e.g. speed, longitude, latitude, orientation) as well as provide the required control elements to run the monitorized vehicles. This kind of software is usually closely bounded to the vehicle that monitorizes due to the heterogeneity of its controllers and telemetry data sending methods.

Seeing that, the main challenge to deal with is to design an adaptable framework that allow us to monitorize and control a variable number of heterogeneous vehicles regardless of their hardware and software.

This goal is achieved in the Versatile Control Center by a software that reads from XML archives the configuration parameters, an adaptable graphic user interface and a simple and adaptable communication structure.

PALABRAS CLAVE

USV (Vehículo de Superficie no Tripulado), UAV (Vehículo Aéreo no Tripulado), UGV (Vehículo de Tierra no Tripulado), Vehículo Autónomo Heterogéneo, Centro de Control, GCS (Estación de Control de Tierra), Monitorización, Control, Ergonomía, Operador, Comunicaciones, Cliente, Servidor, XML, Configurador, Versátil.

KEYWORDS

USV (Unmanned Surface Vehicle), UAV (Unmanned Aerial Vehicle), UGV (Unmanned Ground Vehicle), GCS (Ground Control Station), Unmanned Heterogeneous Vehicle, Monitorize, Control, Ergonomics, Operator, Communications, Client, Server, XML, Configurator, Versatile, Adaptable.

ÍNDICE

ÍNDICE DE FIGURAS

CAPÍTULO 1: INTRODUCCIÓN

En este capítulo trataremos el problema del diseño de un centro de control en base a las necesidades derivadas de la monitorización de un conjunto variable de vehículos no tripulados de naturaleza heterogénea y la ubicación dinámica de sus componentes visuales según los criterios ergonómicos demandados por el operador.

1.1 Motivación

Una estación de tierra suele recoger información sobre el estado de un vehículo monitorizado (p. ej. posición, orientación, nivel de batería), las condiciones del entorno (p. ej. posibles obstáculos, mapas de situación, objetivos, metas alcanzadas y pendientes) e incorporar dispositivos para su control (p.ej. control manual, establecimiento de misiones). Todos estos elementos, cuya presencia es fundamental en cualquier centro de control, y su ubicación en la pantalla cobra especial importancia en casos en los que el uso de múltiples vehículos es necesario para la correcta ejecución de la misión en desarrollo. Esto se debe a que, al crecer el número de vehículos simultáneos a monitorizar y controlar, el operador debe centrar su atención en un número mayor de indicadores, aumentando la dificultad del desarrollo de su trabajo y por lo tanto, aumentando la importancia de un diseño ergonómico y adaptado a sus preferencias.

Aunque es posible encontrar estaciones de tierra comerciales [1] [2] o de código abierto [3] capaces de monitorizar múltiples vehículos y estudios en los que se plantean misiones en las que toman parte varios vehículos (UAV's, Unmanned Aerial Vehicles, vehículos aéreos no tripulados) de forma simultánea [4], no resulta fácil encontrar estudios o estaciones de control que permitan la inclusión simultánea de múltiples vehículos heterogéneos ni la reubicación de sus componentes visuales. Por lo tanto, un software de estas características, que permita la incorporación de UAV's y USV's (Unmanned Surface Vehicles, vehículos de superficie no tripulados) es un campo abierto a la investigación.

Por estas razones, el objetivo de este proyecto es ofrecer un centro de control adaptable a un número variable de vehículos de naturaleza heterogénea que además permita la configuración dinámica de sus componentes visuales. Esto se conseguirá por medio de la integración en nuestro centro de control de tres grandes bloques de componentes: un interfaz gráfico modular, un sistema de configuración y un esquema de comunicaciones flexible capaz de comunicarse tanto con vehículos de cualquier naturaleza como con visores y otros centros de control realizando mínimas modificaciones.

El sistema de configuración, junto al método desarrollado para su implementación, será el encargado de indicarle al centro de control la disposición que deben

mostrar por pantalla los módulos del interfaz de usuario además de configurar ciertos aspectos ligados a las comunicaciones para un correcto funcionamiento del esquema de múltiples vehículos diseñado.

Además, un aspecto importante en misiones complejas como tareas de búsqueda, detección o seguimiento es la inclusión de los algoritmos necesarios para que puedan realizarse de forma coordinada la cooperación de varios vehículos autónomos. Una estación de tierra que incorpore estos algoritmos y permita su uso o configuración de un modo sencillo sería un elemento crucial para el éxito de cualquier misión. Así, estos dispositivos ya no sólo son estaciones de tierra que monitorizan y dan consignas a los vehículos autónomos, sino que toman decisiones complejas para el éxito de la misión. Esto es lo que denominaremos Centro de Control Versátil (CCV).

1.2 Objetivos

El objetivo principal de este proyecto es diseñar un centro de control de tierra versátil y reconfigurable gracias a un sencillo archivo XML (eXtensible Markup Language) en el que el usuario podrá definir los controles (integrando los controladores de los vehículos existentes hasta el momento) y parámetros a monitorizar así como los vehículos a los que pertenecen, tratando de solventar los principales problemas a los que nos enfrentamos con las soluciones existentes en la actualidad ya sean comerciales [1] [2] (p. ej. tamaño y dificultad de transporte, tecnología, coste) o de código abierto [3] [4] (p. ej. versatilidad limitada en configuración de interfaz de usuario, disponibilidad de controles para un solo tipo de vehículo).

Es necesario, además, antes de construir un centro de control que permita la coordinación de los diferentes vehículos no tripulados, definir con la máxima precisión posible aspectos como el tipo de datos y comunicaciones de cada uno de ellos, los mapas del entorno, los tipos de enlace de datos y la velocidad de transferencia, etc. Por tanto, un objetivo de este Trabajo de Fin de Máster es examinar qué tipos de estaciones de tierra existen actualmente, que vehículos autónomos podrán ser gestionados y qué tipos y protocolos de comunicación deben tenerse en cuenta. Con esta información se realizará el diseño más adecuado para conseguir los objetivos propuestos y se desarrollará un prototipo que demuestre cómo construir el Centro de Control Versátil, que permitirá la gestión de diversos vehículos y su comunicación con el centro de control mediante diferentes protocolos. Todo ello de forma que sea configurable por el usuario. De este modo se podrán escoger diferentes controles, planificadores, alarmas, etc.

Para implementar el prototipo del Centro de Control Versátil se seguirán los siguientes pasos:

1.- Análisis de diferentes estaciones de tierra, atendiendo al número y diversidad de vehículos que pueden controlar, tipo de comunicación, disposición de los elementos de monitorización, etc.

2.- Implementación de un centro de control reconfigurable. Aunque sea un prototipo debe ser diseñado de modo que pueda servir de base para un futuro centro de control más complejo y versátil, es decir, que su diseño permita ampliaciones y mejoras futuras. El prototipo deberá contemplar la posibilidad de las siguientes opciones:

- Mostrar la posición de los vehículos sobre mapas del terreno. Se estudiará la posibilidad de utilizar imágenes de Google Earth o mapas de uso libre (base de datos OpenStreetMaps [5] mapas o del Instituto Geográfico Nacional [6]).
- Mostrar la información del estado de los vehículos elegida en el archivo XML responsable de configurar, para cada misión, la pantalla general de monitorización del centro de control.
- Actuar como servidor de información a programas de clientes autorizados, de modo que permita mostrar en otros equipos/formatos los datos de posición de los vehículos monitorizados.
- Definir, mediante archivos XML, diversas alarmas sobre variables monitorizadas para avisar al operador si se alcanzan unos determinados valores y/o modificar automáticamente el comportamiento de los vehículos durante el desarrollo de la misión.
- Incorporar el software de control de los vehículos que actualmente están siendo utilizados en el Centro de Control Versátil

1.3 Estado del Arte

Una estación de tierra es un dispositivo que permite la monitorización (p. ej. posición sobre un mapa, orientación, nivel de batería, velocidad, altitud) de vehículos no tripulados ya sean aéreos (UAV's), terrestres (Unmanned Ground Vehicles – UGV's) o de superficie (USV's) además de proporcionar las herramientas necesarias para su control ya sea automático (p. ej. rutas o misiones previamente programadas) o manual (p. ej. parada de emergencia, cambio de orientación, velocidad).

Habitualmente el diseño de estos dispositivos se enfoca en un tipo de vehículo concreto (UAV, UGV o USV), ofreciendo una serie de funcionalidades y herramientas que, si bien en algunos casos pueden utilizarse para modelos heterogéneos (p. ej. indicadores como un velocímetro o una brújula o control de orientación podrían resultar útiles para casi cualquier tipo de vehículo), pueden resultar totalmente prescindibles en otros (p. ej. indicadores como un horizonte artificial resultaría imprescindible en un UAV pero podría resultar innecesario en un UGV). Este diseño tan ligado a un tipo concreto de vehículo suele hacer que el centro de control resulte inservible o poco útil cuando nos referimos a él en términos de reutilización. A esto habría que sumarle otros factores como un más que probable cambio en el protocolo de comunicaciones. Plantearse realizar estos cambios suele ser difícil ya sea por falta de documentación sobre el programa, por el hecho de que si el código no es abierto no se podrá realizar modificaciones en el software, o por resultar estos cambios demasiado costosos.

También podemos necesitar realizar una misión más o menos compleja en la que deban intervenir vehículos de naturaleza heterogénea. Aunque podemos encontrar estaciones de control que permitan monitorizar varios vehículos de forma simultánea, habitualmente sólo posibilitan el control de vehículos de un mismo tipo, ya sea UAV, UGV o USV [3] [7] [4]. Existe una excepción en la que los terminales ofrecen la recepción directa de video desde vehículos heterogéneos de forma simultánea en varios equipos (independientemente del tipo de estación de tierra) a costa de prescindir totalmente de elementos de control y variables de monitorización del aparato, que estarían controlados por la propia estación en un software separado [8]. En el caso de plantearse realizar esta tarea, a no ser que se cuente con una estación de control específicamente diseñada al efecto, el usuario se verá obligado a contar con varias estaciones de control, aumentando los recursos necesarios y pudiendo requerir la inclusión de un mayor número de personas para llevar a cabo un trabajo que, con un diseño adecuado, podría realizar un solo operador.

Entre las estaciones de control que se encuentran disponibles en el mercado podemos diferenciar dos grandes grupos, las comerciales o militares (ver Figura 1), y las de código abierto (ver Figura 2). Podríamos distinguir las primeras por ser más avanzadas tecnológicamente (principalmente a nivel hardware) debido a una mayor disponibilidad de recursos económicos. Aunque no es fácil acceder a especificaciones e información de este tipo de estaciones sí podemos encontrar algún ejemplo como los de la compañía UAS [9] con precios de software que oscilan entre 2400€ y 6900€ (dependiendo de funcionalidades disponibles) y precios para hardware que se sitúan en 16900€ por una estación de control portátil

que incluiría un sencillo ordenador portátil, algunos controles y un maletín de protección de categoría IP65 [7]. Un ejemplo puede verse en la Figura 1. Otros tipos de estaciones de control portátiles similares pueden encontrarse disponibles por parte de otros fabricantes como UCON o UAV Factory [10] [11]. Aunque existen este tipo de estaciones de control portátiles, en el mercado comercial habría que destacar las que cuentan con un tamaño considerablemente mayor, Estaciones de Control Tácticas o de tierra, para cuyo transporte sería necesario un camión o similar y deberían dejarse fijas en un punto [12] [13] [14] [15] [16] (ver los ejemplos en Figura 1). Este tipo de Estación de control destaca por un hardware que permite un mayor radio de control y, en general, unos recursos tecnológicos difícilmente alcanzables en el ámbito de la investigación. No obstante, este tipo de estación de control suele ir estrechamente ligada al tipo de vehículo monitorizado, pudiendo presentar algunas de las desventajas anteriormente señaladas.

Figura 1: De izq. a der., Estación de Control Portátil, Estación de Control Militar y Est. de Control Comercial

En el otro grupo, las Estaciones de Control de código abierto [3] [17] (ver Figura 2), podemos encontrar en su mayoría un software que el usuario deberá instalar en un Ordenador Portátil y que proporciona las herramientas necesarias para la monitorización y/o control del vehículo. Se trata de unas estaciones de control bastante más accesibles (ya que tan sólo requieren un ordenador portátil para ejecutar su software, que es habitualmente gratuito), y razonablemente flexibles (ya que gracias a que su código es abierto, un usuario con los conocimientos apropiados y la dedicación necesaria podrá adecuarlo a sus necesidades concretas). Las estaciones de control de esta rama cuentan por tanto con una mayor versatilidad y accesibilidad (especificaciones, documentación y manuales disponibles para todo el público) además de una mayor portabilidad, ya que lo usual es contar con un ordenador portátil de unas dimensiones y peso razonables, y una necesidad de inversión mucho menor. Por contra, parten con desventaja respecto al mencionado nivel tecnológico, con unas comunicaciones de menor alcance, y menor robustez y seguridad.

Figura 2: Centro de Control de Código Abierto

En cualquier caso, el nivel de versatilidad que presenta este tipo de sistema de control es menor que el ofrecido en el diseño que se propondrá en este trabajo ya que, en ningún caso se plantea la posibilidad de reconfigurar los indicadores, controles y alertas (visuales o sonoras) adecuados a las características de cada vehículo, misión y operador, hecho que puede influir (negativamente en caso de una disposición incorrecta o positivamente en caso de una disposición adecuada) en el rendimiento del mismo y por tanto en el correcto desarrollo de la misión, como han demostrado algunos estudios al respecto [18].

Una vez planteadas las dificultades anteriores, con las que nos hemos topado en la mayoría de estaciones de control a las que podemos acceder en el mercado, puede comenzar a intuirse el propósito de este proyecto. Por un lado se pretende diseñar un centro de control que sea funcional para cualquier tipo de vehículo no tripulado, implementando los controles necesarios para todos ellos (tanto los comunes como los específicos) y permitiendo al usuario, mediante una sencilla configuración, elegir los que sean útiles en cada caso basándose no sólo en las características del vehículo en cuestión y las necesidades derivadas de la misión sino también en sus preferencias personales. De esta forma, gracias al archivo de configuración, en unos segundos puede cambiarse tanto la disposición como la cantidad o naturaleza de los indicadores o controles pudiendo alternarlos de forma que el centro de control sea funcional para un tipo de vehículo u otro. Este archivo de configuración, un sencillo documento en formato XML, permitirá alteraciones de varios tipos sobre el interfaz de usuario. Algunos más sencillos, se basarán en criterios ergonómicos del usuario (p. ej. la situación dentro del interfaz de un indicador a izquierda o derecha o qué indicadores críticos deben aparecer en primer plano o surgir como una alarma cuando se alcancen ciertos valores). Otros más complejos se realizarán de forma transparente al usuario mediante unas directivas básicas que se le darán al programa desde dicho archivo de configuración (p. ej. cambios en la codificación, protocolos, naturaleza o tipo de variables y demás cambios derivados del tipo de vehículo a visualizar se llevarán a cabo automáticamente con tan sólo indicarlo en una variable como "UAV", "UGV" o "USV"). El centro de control también podrá reconfigurarse el tipo y número de vehículos incluidos en el sistema (p. ej. cambio de misión) de forma rápida y sencilla cargando un nuevo XML con la configuración necesaria de cada nuevo vehículo.

Este tipo de diseño permitirá que se logren una gran cantidad de configuraciones diferentes en el apartado visual lo que sumado a la flexibilidad respecto a número y tipo de vehículos compatible y tipo de comunicaciones (p. ej. recepción desde uno o varios emisores simultáneamente independientemente de la naturaleza o codificación de los mismos, recepción simultánea de un mismo vehículo desde dos estaciones de tierra que incluso podrían tener los controles e indicadores en posiciones distintas, o configurar el centro de control como emisor para un tercer equipo más ligero como un dispositivo android o tablet pc), permite alcanzar una versatilidad que, al menos hasta donde alcanza nuestro conocimiento, no se ha visto anteriormente en otras estaciones de control.

Por lo tanto, a pesar de que el centro de control que nos proponemos desarrollar cuenta con algunas desventajas respecto a ciertas estaciones de control de las anteriormente mencionadas (p. ej. respecto al radio de alcance en las comunicaciones), también podrán encontrarse notables ventajas como versatilidad en la disposición de elementos en la pantalla, posibilidad de configuración variable de la cantidad de vehículos independientemente de su naturaleza (homogénea o heterogénea), flexibilidad en cuanto a la arquitectura, que permitirá varios tipos de configuración entre emisores y receptores, y simplificación notable en la conexión con nuevos vehículos o dispositivos determinando un protocolo sencillo adaptable casi a cualquier plataforma y potenciado con un archivo de configuración XML.

1.4 Organización del Trabajo de Fin de Máster

La organización de la memoria de este Trabajo de Fin de Máster es la siguiente:

En el Capítulo 2 se presenta el esquema general de funcionamiento del Centro de Control Versátil junto con algunos ejemplos ilustrativos además del tipo de comunicaciones elegido junto a la estructura Cliente-Servidor que dota al sistema de una gran flexibilidad de configuración.

En el capítulo 3 se detallarán los módulos independientes de visualización de datos y las diferentes configuraciones que pueden surgir en base a la ubicación de los mismos según las preferencias del operador.

En el Capítulo 4 se explicará todo lo relacionado con el archivo de configuración XML, la especificación de su DTD (Document Type Definition, Definición de Tipo de Documento), significado de cada uno de sus elementos y atributos y forma de utilizarlo para diseñar un interfaz de usuario.

En el Capítulo 5 se analizarán los ensayos y pruebas llevados a cabo sobre el Centro de Control Versátil, partiendo con ensayos previos sobre algunos de sus componentes y terminando con pruebas generales de funcionamiento sobre el terreno con USV's reales.

En el Capítulo 6 detallaremos las conclusiones tras el desarrollo del proyecto, objetivos cumplidos y finalizaremos describiendo las ideas surgidas para mejorar y ampliar el Centro de Control Versátil además de posibles pruebas y estudios realizables.

En el apartado de Apéndices Explicaremos dos de las herramientas de desarrollo más importantes que se han utilizado para este proyecto. En primer lugar hablaremos de OpenStreetMaps [5] y JMapViewer [19], que proporcionan mapas de código abierto y una serie de clases para manipularlos e integrarlos en un proyecto Java. Además comentaremos brevemente las modificaciones realizadas para ajustar JMapViewer a las necesidades del Centro de Control Versátil. En segundo lugar describiremos brevemente la herramienta JDOM [20], una herramienta que permite la lectura y manipulación de archivos XML, y cómo se ha utilizado en nuestro proyecto.

CAPÍTULO 2: DISEÑO GENERAL Y COMUNICACIONES

En la primera parte de este capítulo explicaremos el diseño general que tendrá el proyecto y sus diferentes configuraciones, mientras que en la segunda parte detallaremos el tipo de comunicaciones elegidas para realizar el enlace entre el Centro de Control Versátil y los vehículos.

2.1 Diseño General

A continuación explicaremos la estructura general del diseño del proyecto. Partiremos de un esquema general de componentes y conexiones entre ellos, explicaremos a grandes rasgos los pasos principales para comunicarlos, además de por qué el diseño elegido ofrece gran versatilidad.

2.1.1 Estructura General

En primer lugar comenzaremos dando una breve descripción de la del centro de control. Podemos ver sus componentes principales en la Figura 3.

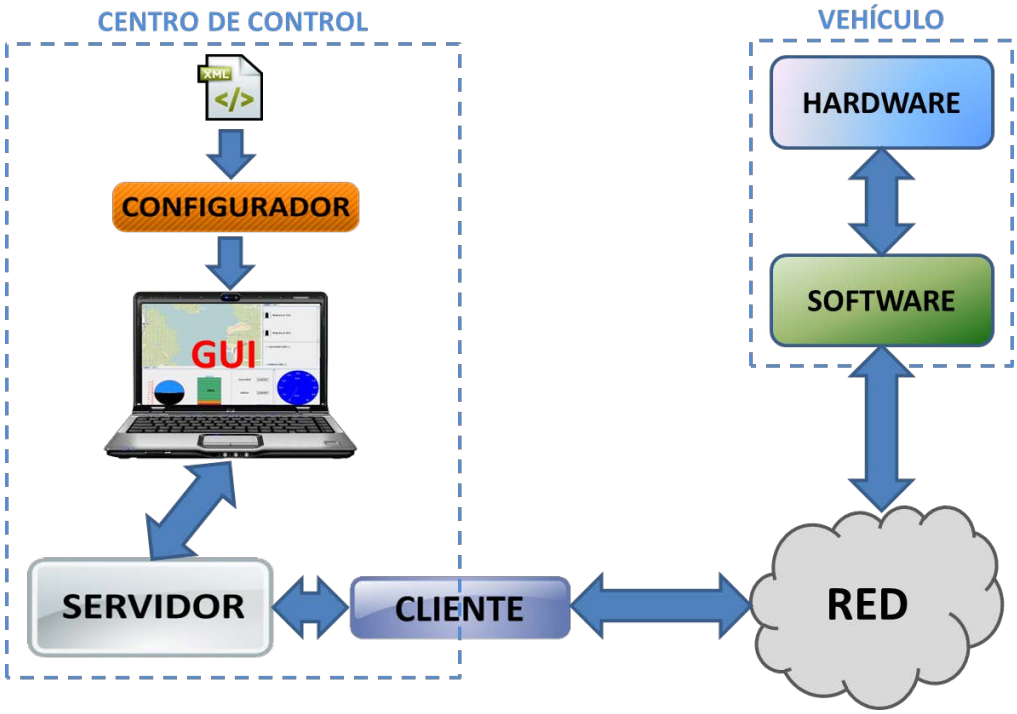

Figura 3: Componentes de la Estructura General

- **Configurador XML**

  Se trata de un módulo encargado de realizar la apertura, lectura y tratado de datos que aparecen en un documento XML de configuración para el Centro de Control. Dicho documento deberá estar previamente preparado en base a unas directrices concretas para una correcta configuración de la Centro de Control.

  El configurador podrá leer uno o varios documentos XML. En cada uno de ellos representará la configuración que desea cargarse para cada uno de los vehículos monitorizados.

  También es el módulo encargado de leer del documento XML los datos necesarios para que se realice correctamente el enlace vehículo – Interfaz de Usuario, validando ciertos datos necesarios para su correcto funcionamiento.

- **Interfaz de Usuario(GUI)**

  Es uno de los componentes más importantes en un centro de control ya que es el elemento que mostrará al usuario toda la información relevante y con el que éste interactuará a la hora de monitorizar y/o controlar los vehículos no tripulados.

  Podrá reconfigurarse en base a necesidades del vehículo, la misión y preferencias personales.

  Se encargará de recibir las actualizaciones periódicas de datos referentes a los vehículos y de mostrarlas como se haya especificado.

- **Servidor**

  Al iniciar el programa principal del Centro de Control, se iniciará un Servidor encargado de atender las peticiones de conexión por parte de los Clientes (alojados en el hardware del vehículo o en el PC del centro de control).

  Al recibir una petición de apertura de comunicaciones, creará un canal dedicado para comunicarse con el vehículo, comprobará los datos de control recibidos y, si son correctos, enviará una confirmación al Cliente.

  Si los datos son incorrectos, se cerrará el canal dedicado con ese vehículo, se enviará una señal de cierre al cliente y, en su caso, un mensaje que indique por qué son incorrectos los datos de telemetría para su posterior corrección y reenvío.

  Se encargará también de recibir los datos enviados periódicamente por el vehículo y proporcionárselos a la interfaz de usuario para su tratamiento.

  También podrá encargarse del reenvío de datos a un tercer terminal receptor que cuente con un visor o interfaz de usuario, de modo que permita replicar la señal recibida para su visualización en múltiples dispositivos.

- **Cliente**

  Se trata de un programa ligero conectado al software de control del vehículo no tripulado.

  El cliente podrá encontrarse en el PC emisor o receptor para una mayor flexibilidad en las comunicaciones. Actuará como middleware (software intermedio) entre un Servidor y un Vehículo gracias a un interfaz de programación en el que podrán encapsularse las primitivas de adquisición de datos de telemetría propios de cada vehículo y de esta forma enviar los datos recibidos de las mismas en el formato correcto de forma transparente al usuario.

  Se encargará de realizar la petición de conexión al Servidor residente en el equipo con el Centro de Control. Una vez realizada la petición, enviará los datos de control y, si éstos son correctos, recibirá una confirmación de que el canal de comunicaciones está abierto.

  Una vez abierto el canal, se encargará de enviar periódicamente los datos que se desee visualizar en el interfaz de usuario.

- **Vehículo No Tripulado**

  Vehículo que se desea monitorizar y/o controlar de forma remota desde nuestro Centro de Control.

  En este caso puede tratarse de un vehículo aéreo (UAV), terrestre (UGV), de superficie (USV) o cualquier combinación de los mismos.

  Tendrán incorporados los sensores y dispositivos necesarios para generar los datos de telemetría y controlar el vehículo de forma remota.

  En algunos casos el vehículo contará con un PLC o terminal Hardware acoplado (p. ej. Raspberry Pi o Arduino) que actuará como enlace con los equipos en los que se ejecuten el controlador o el Centro de Control Versátil.

2.1.2 Versatilidad y Flexibilidad para el Usuario

Como podremos observar en los distintos esquemas que se presentarán a continuación, el diseño de la arquitectura tanto en el aspecto físico (PC's, Tablets, Vehículos) como en las múltiples combinaciones de configuración Cliente – Servidor para las comunicaciones, ofrece una versatilidad difícilmente alcanzable por alguna de las estaciones de control disponibles en el mercado.

El primer factor diferenciador es la posibilidad que ofrece de conectar el centro de control a un número variable de vehículos sin necesidad de realizar cambios en la estructura Hardware (p. ej. emisores, receptores, PC's), ni en la estructura de comunicaciones, que se reconfigurará de forma automática en función de las necesidades del esquema elegido, ni en el código del software del Centro de Control Versátil.

El segundo factor que ofrece una diferencia considerable es la posibilidad de que los vehículos incluidos para su control y monitorización puedan ser de naturaleza heterogénea, lo que es muy poco común en las estaciones de control actuales.

Otra de las grandes ventajas que podemos encontrar en el diseño estructural del Centro de Control Versátil es la posibilidad de configurar uno o varios Centros de Control, e incluso algún visor para seguimiento remoto, pudiendo configurar los centros dependiendo de las necesidades para uno o varios operadores.

Esto se logra gracias a una arquitectura versátil que permite reconfigurar tanto a nivel de comunicaciones (tanto en número de vehículos enlazados como configuraciones y número de receptores ya sean centros de control o simples visores) como controles logrando una total flexibilidad en su utilización ya sea por uno o varios operadores. Este diseño arquitectónico, potenciado con un archivo de configuración XML, permite que un usuario sin unos conocimientos específicos de programación, comunicaciones etc. pueda configurar de forma sencilla, transparente y automática el Centro de Control Versátil de la forma que más se ajuste a sus necesidades.

2.1.3 Ejemplos de Funcionamiento

A continuación mostraremos los ejemplos de los esquemas básicos de funcionamiento en los que se basa la arquitectura del Centro de control Versátil.

Comenzaremos con un esquema simple con un solo vehículo y posteriormente desarrollaremos algunos más complejos.

2.1.3.1 Esquema 1: Un Centro de Control, Un Vehículo

En este esquema podemos observar la configuración arquitectónica más simple para la que puede usarse el Centro de Control Versátil, un solo vehículo con un solo Centro de Control.

El configurador solo deberá cargar un documento XML con la configuración relativa al vehículo (en este caso un UAV) y el servidor abrirá un único canal de comunicaciones con el cliente alojado en el Hardware de enlace del UAV si lo tuviese (una Raspberry Pi en el caso del ejemplo de la Figura 4) que proporcionará los datos que se transmitirán al interfaz de usuario. De no contar el vehículo con dicho hardware de enlace o tener capacidad éste de ejecutar el cliente, habría que hacerlo en el equipo en el que esté el centro de control. Podemos ver un ejemplo de este tipo de este esquema en la Figura 4.

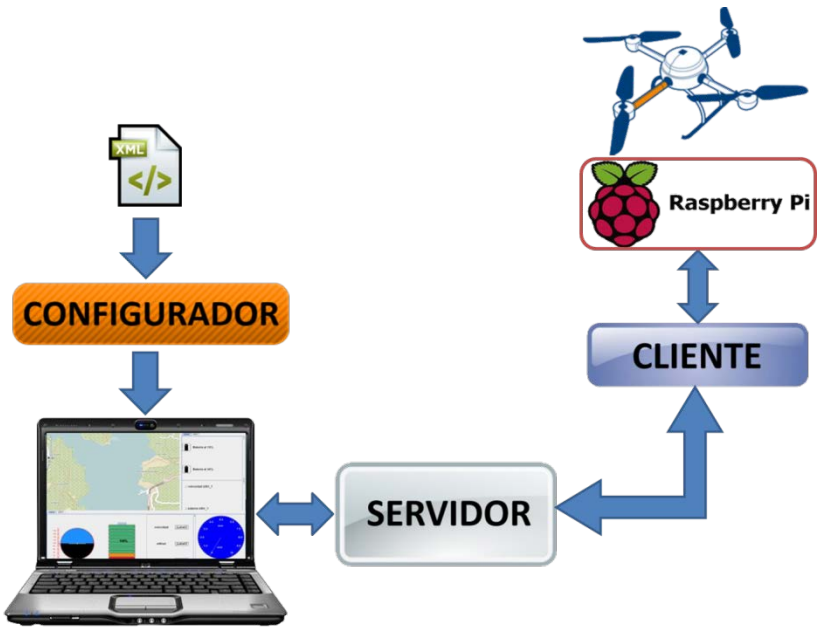

Figura 4: Un Centro de Control, Un vehículo

2.1.3.2 *Esquema 2: Un Centro de Control, Varios Vehículos*

El siguiente esquema aumenta su complejidad agregando al caso anterior dos nuevos vehículos: un USV y un UGV.

En este caso deberán cargarse 3 documentos XML, uno por vehículo, en los que figurarán la configuración y datos necesarios para monitorizarlos.

Se abrirán 3 canales de comunicación independientes por los que los controladores software de los vehículos, a través de su Cliente incrustado, enviarán los datos al interfaz de usuario del Centro de Control. Ver esquema en la Figura 5.

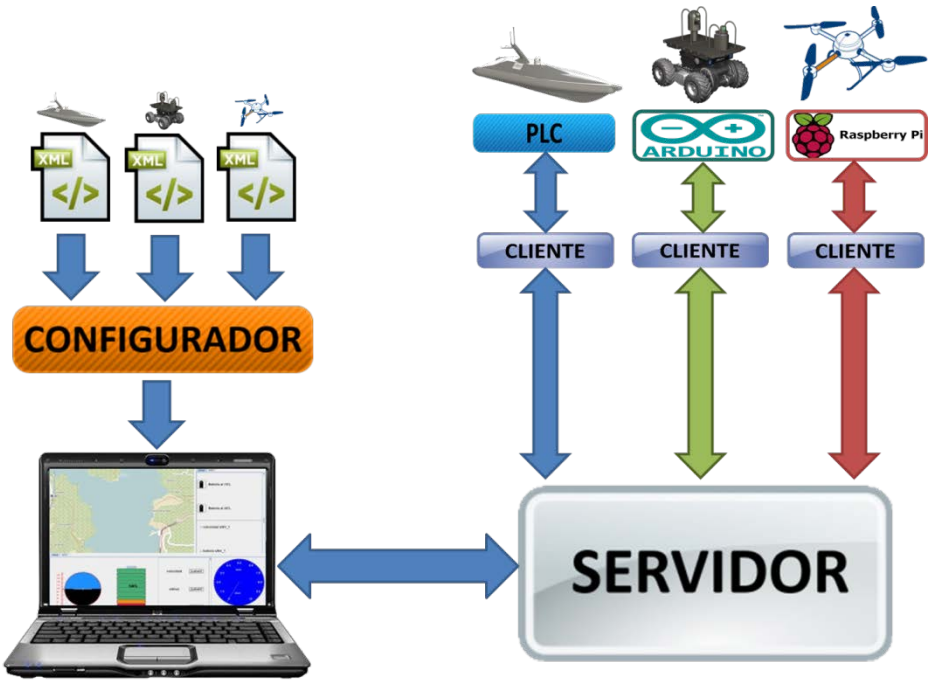

Figura 5: Un Centro de Control, Tres Vehículos

2.1.3.3 Esquema 3: Dos Centros de Control, Varios Vehículos

En este caso el esquema representa la misma estructura de vehículos que el ejemplo anterior, diferenciándose en el uso de dos interfaces de usuario.

Ambas pueden estar configuradas exactamente igual si así se desea (cargando los mismos archivos XML en las dos) o de forma personalizada si se considera oportuno. Esto dota de una enorme flexibilidad a la estructura, ya que si se dispone de dos operadores para realizar el seguimiento, cada uno puede centrarse en aspectos distintos de la misión, en los mismos, o en algunos comunes y otros distintos. También podría cargarse un único vehículo en alguna de las estaciones si así se desea, de forma que un operador se centre en dicho vehículo. Podemos ver un ejemplo de este esquema en la Figura 6.

Es importante destacar que en el caso de contar con más de un centro de control con más de un operador funcionando de forma concurrente, sería necesario implementar un sistema de prioridad con el objetivo de evitar posibles conflictos en el control (ver sección Trabajo Futuro).

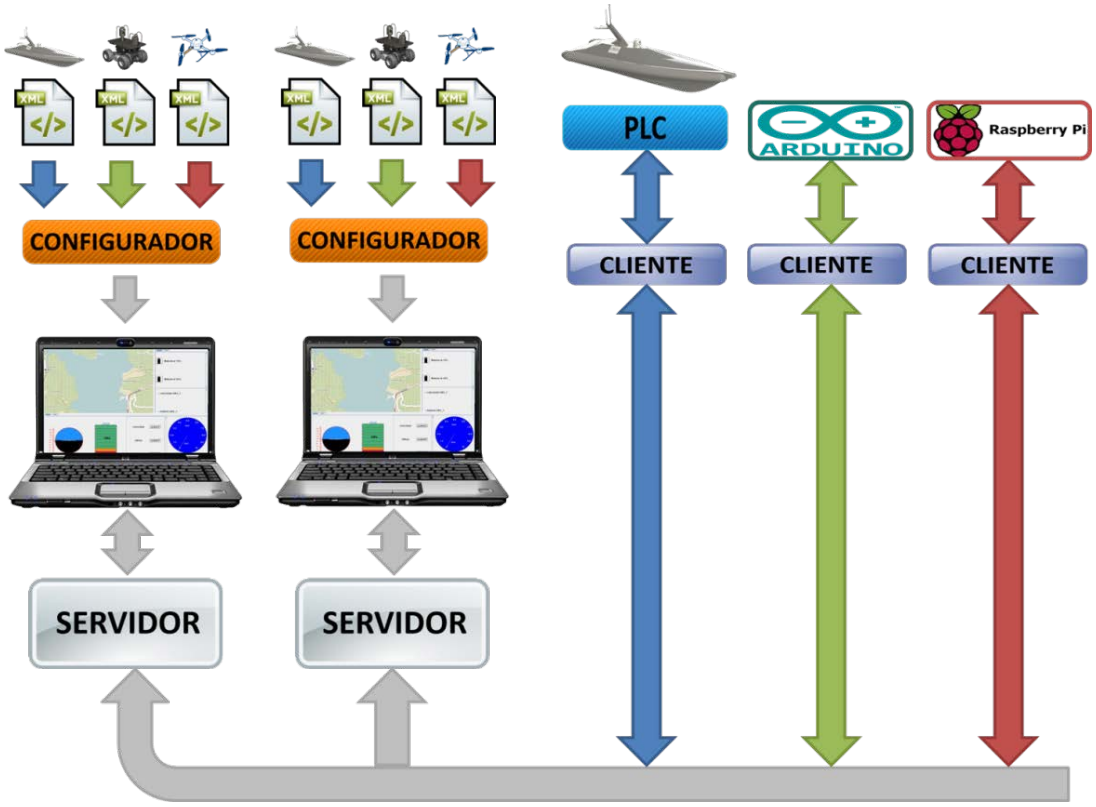

Figura 6: Dos Centros de Control, Varios Vehículos

2.1.3.4 Esquema 4: Dos Centros de Control, Un Visualizador, Varios Vehículos

Por último mostraremos un ejemplo en el que la estructura es muy similar a la del esquema anterior, con la salvedad de que se incluye un tercer visor que se carga en un terminal ligero (Tablet PC, eventualmente con un sistema operativo distinto al del resto de ordenadores del sistema) que en este caso sólo recibe datos para la visualización de todos o alguno de los vehículos.

Esta posibilidad aporta nuevas opciones, configurando el servidor a modo de repetidor, tomando los datos que recibe de alguno o todos los vehículos para uno de los visores y reenviándolos por otro canal hacia el tablet PC. Consultar ejemplo de esquema en Figura 7.

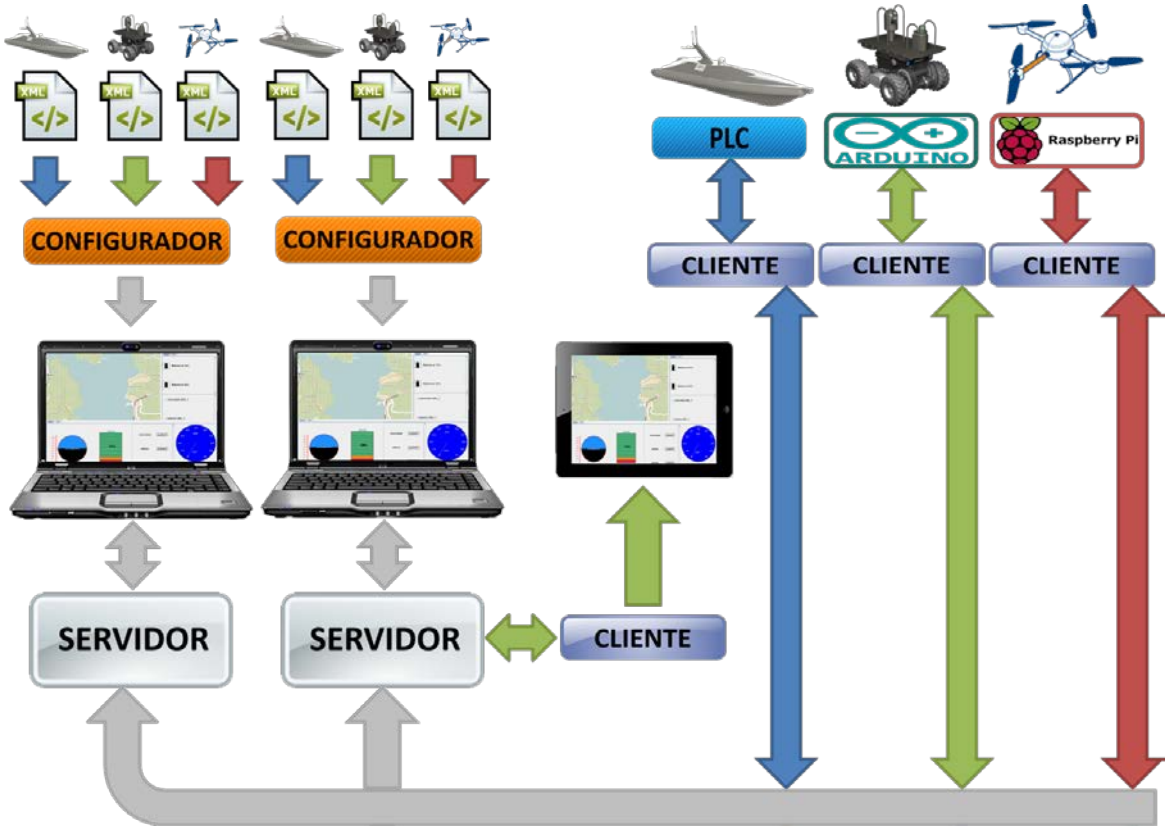

Figura 7: Dos Centros de Control. Un Visor. Varios Vehículos

2.2 Tipos de Comunicaciones

La naturaleza heterogénea de los vehículos con los que el Centro de Control Versátil debe comunicarse requiere el soporte de múltiples tipos de comunicación.

A la hora de decidir qué tipos de conexiones utilizar en los ensayos del Centro de Control Versátil hemos tratado de elegir un tipo de comunicaciones lo más standard posibles, adecuada a los tipos de vehículos disponibles para dichos ensayos y que además sea lo más reutilizable posible, es decir, que permita la conexión de vehículos heterogéneos con la mayor facilidad posible.

Por esta razón, se han elegido conexiones TCP ya que permiten comunicar una gran cantidad de equipos independientemente de su plataforma y arquitectura mediante conexiones habituales como pueden ser wi-fi o red cableada.

2.2.1 Comunicaciones TCP

Para las comunicaciones de este proyecto se ha elegido una conexión por sockets tipo TCP, protocolo orientado a conexión en el que las máquinas que se comunican envían información en forma de datagramas.

Los sockets son un sistema de comunicación entre procesos de diferentes máquinas de una red, puntos de comunicación por los cuales los procesos pueden enviar o recibir información. Los sockets pueden utilizarse para establecer conexiones tipo UDP o, como en este caso, tipo TCP. El paquete java.net [21] proporciona las primitivas necesarias para realizar de forma sencilla esta conexión.

La estructura básica cliente-servidor de una comunicación por sockets consta de dos clases, **ServerSocket** que hace las veces de servidor, abre un puerto y queda a la espera de recibir peticiones de conexión y la clase **Socket**, que actuará como cliente. Para realizar la conexión, se deberá proporcionar al cliente la IP del equipo al que se debe conectar y el puerto por el que el servidor espera las peticiones. Una vez recibida y aceptada la conexión, se abrirá un canal por el que ambos enviarán y recibirán flujos de datos.

Se ha elegido este tipo de conexión debido a varios factores. En primer lugar se trata de un tipo de conexión muy extendido y por lo tanto disponible para cualquier plataforma y lenguaje de programación, lo que ayuda a conseguir la versatilidad necesaria a la hora de conectar el software del proyecto con cualquier tipo de vehículo independientemente del hardware y software del que esté formado. En segundo lugar, se trata de una conexión relativamente segura y que permite enviar confirmaciones una vez recibido un mensaje, lo que nos permite realizar una verificación de variables, vehículos disponibles y sus características en el interfaz de usuario antes de aceptar la conexión para evitar fallos críticos y posibles intrusiones de sistemas que no cuenten con los requisitos necesarios al enviar los flujos de datos. Esto se realiza mediante una lectura previa del documento de configuración XML, el cliente envía junto a la petición inicial el identificador de vehículo, tipo y una lista de las variables mapeadas de forma que si éstas no coinciden exactamente con las que se han introducido en el documento XML del Servidor, la conexión se rechazará y se anunciará al cliente el motivo del rechazo. Otro de los motivos es la escalabilidad que proporciona al sistema este tipo de conexión, ya que con una sencilla configuración se permite un número

teóricamente ilimitado de conexiones. Se ha configurado un ServerSocket que gestiona las peticiones de conexión y al aceptar una de ellas, abre un hilo con un canal específico para esa conexión con el cliente. De esta forma cada conexión estará disponible en todo momento para un vehículo que puede estar enviando datos críticos. Por último la estructura cliente-servidor dota de una mayor flexibilidad a la hora de configurar las conexiones, dependiendo de dónde se ubique el cliente, como explicaremos más adelante.

2.2.2 Flexibilidad Estructura Cliente-Servidor

La estructura cliente-servidor establecida para realizar las conexiones del Centro de Control Versátil se ha diseñado para poder variar lo más posible las comunicaciones con los vehículos sin necesidad de alterar el código (o alterándolo mínimamente, cambiando puerto e IP) al modificar la ubicación del cliente (que puede ejecutarse en el equipo en el que se encuentra el centro de control, alojado en el Hardware del vehículo o incluso en un equipo intermedio que retransmita los datos recibidos a un tercero).

A continuación explicaremos brevemente dos de los esquemas más probables.

2.2.2.1 Servidor y Cliente en el Mismo Equipo

La primera modalidad de conexión y quizá la más sencilla es la que aloja en el mismo PC el interfaz del Centro de Control Versátil, el servidor y el Cliente.

En este caso, el cliente se comunica con el servidor en un lazo local (**localhost** o IP 127.0.0.1) por el puerto que se le indique y con el software controlador del vehículo externo por medio del interfaz preparado al efecto, que mediante unos métodos que deberán sobreescribirse para cada vehículo por parte del usuario, invocará a las primitivas disponibles para la petición o recepción de datos enlazados a las variables descritas y que tratará el socket receptor que ha habilitado el servidor para recibir los flujos de datos por el canal de ese vehículo.

La conexión cliente-servidor en **localhost** presenta la ventaja de reducir los retardos entre emisión y recepción, además de minimizar posibles errores debidos a fallos en comunicaciones (fallos de red, pérdidas de cobertura…etc.). Como desventaja podríamos señalar el aumento de carga de trabajo del PC en el que esté alojado ya que sobre éste recae todo el trabajo de computación (interfaz gráfico + Servidor + Cliente/Clientes). Sin embargo al tratarse de clientes relativamente sencillos y ligeros no es habitual que se presenten problemas derivados de este aumento de carga como ralentizaciones o bloqueos, cuando se usan equipos con la capacidad de computación y memoria actuales.

Podemos observar un esquema de estas características en la Figura 8.

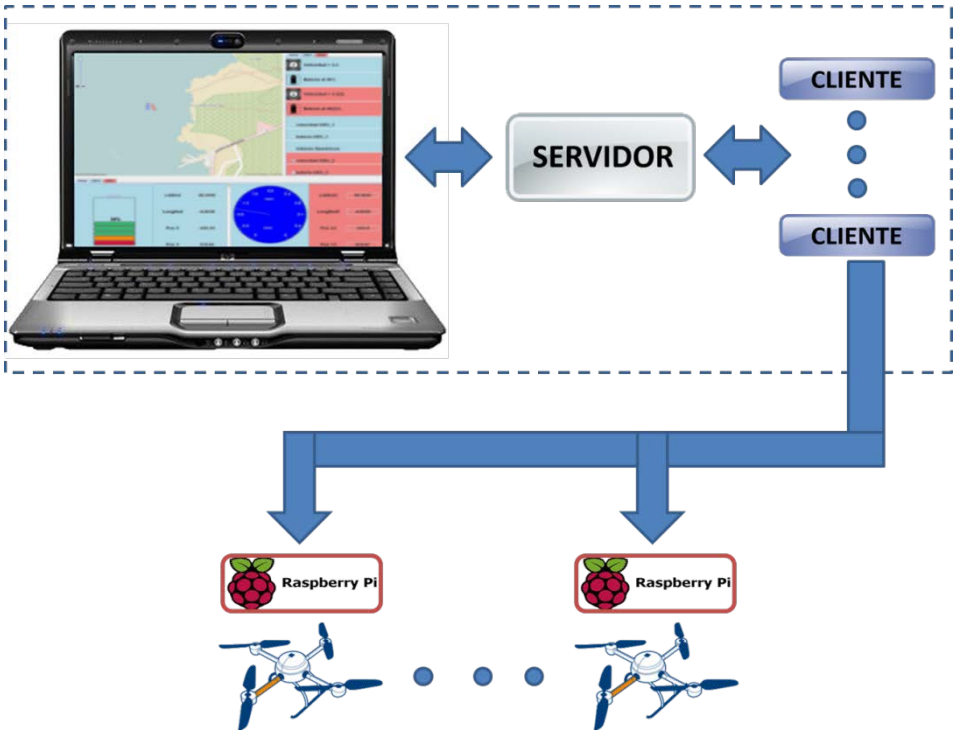

Figura 8: Servidor y Cliente en Mismo Equipo

2.2.2.2 *Servidor y Cliente en Equipos Independientes*

En este modo de conexión, el servidor permanece en el mismo equipo que el interfaz del Centro de Control Versátil mientras que el cliente se alojará en un equipo externo, ya sea intermedio o el propio equipo que aloja el software de control del vehículo autónomo. La necesidad de uso de un equipo intermedio dependerá de la capacidad del equipo de control del vehículo para alojar el software del cliente. Además, en el caso de elegir el esquema con PC intermedio, la configuración puede contar con un cliente en un PC, varios clientes en un PC o varios PC's con un cliente en cada uno. Podemos ver un esquema de comunicaciones con el Cliente alojado en el Hardware del Vehículo en la Figura 9, mientras que podemos observar diferentes configuraciones del esquema con el Cliente en equipos intermedios en la Figura 10.

En este caso el Cliente se realizará la petición de conexión al servidor, que en caso de recibir datos correctos abrirá un canal específico de comunicaciones y con el software de control del vehículo ya sea para esperar la recepción de datos o para realizar las solicitudes de lectura de los mismos.

La estructura del servidor o del cliente no cambiarán ya que las operaciones a realizar serán las mismas, cambiando sólo la IP y puerto de conexión (cambiando, por ejemplo, una IP concreta por **localhost**).

Este caso presenta la ventaja de la reducción de la carga de trabajo debido al reparto en varios equipos del software (uno con el interfaz gráfico y el servidor, y otro u otros con los clientes) pero podría darse un aumento del retardo entre emisión de los datos y recepción de los mismos. Este retardo es mínimo si se trata de una red dedicada ya que la velocidad de una conexión vía wi-fi o cable a día de hoy son como mínimo varios Mbits/s y los datos enviados son de un tamaño muy limitado. En el caso de transmisiones remotas vía internet, el retardo podría crecer

debido a la cobertura y velocidad ofrecida por el operador además de la distancia que deba ser recorrida. A pesar de estos factores, las comunicaciones deberían producirse con una fluidez debido a, como ya hemos comentado, el tamaño de los datos enviados y las velocidades de conexión actuales.

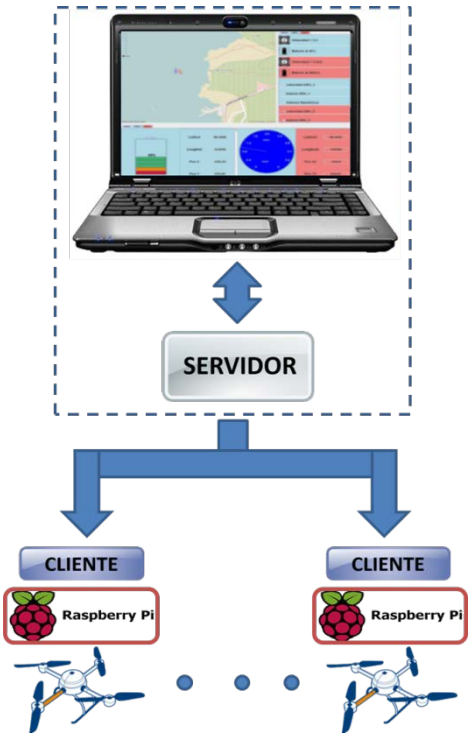

Figura 9: Clientes en Hardware del Vehículo

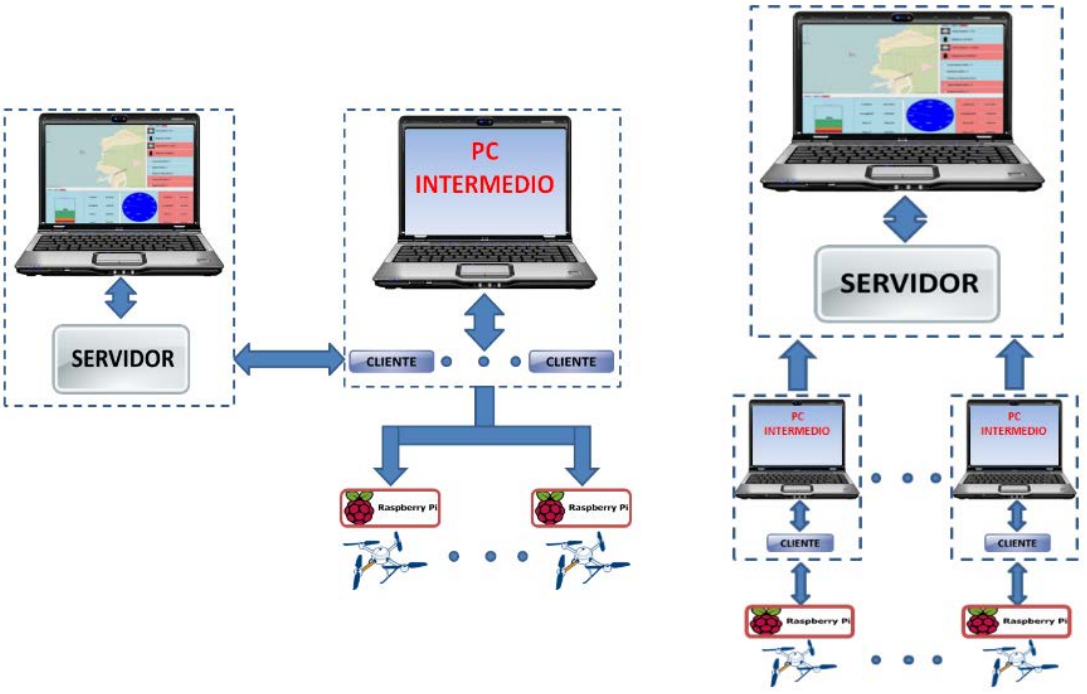

Figura 10: Cientes en Equipos Intermedios, Servidor en Centro de Control

CAPÍTULO 3: INTERFAZ DE USUARIO

En este capítulo describiremos la interfaz de usuario o GUI (Graphic User Interface) del Centro de Control Versátil. Se trata de un apartado clave en el diseño del proyecto, ya que la posibilidad de reconfigurarla y adaptar sus indicadores es lo que nos permite insertar en el sistema cualquier tipo de vehículo autónomo.

Comenzaremos describiendo brevemente la estructura general de la GUI, sus principales elementos contenedores, su función en la estructura general y que tipo de elementos pueden contener. En la siguiente sección de este capítulo describiremos las cuatro configuraciones principales que pueden darse en base a la colocación de estos elementos contenedores y acabaremos analizando los principales elementos indicadores elaborados hasta el momento para la demostración del Centro de Control Versátil.

3.1 Estructura General de la GUI

La interfaz de usuario es una parte esencial para cualquier centro de control, ya que es la parte del software encargada de mostrarle al operador la información sobre los vehículos monitorizados.

Para ello es necesario realizar un diseño que tenga en cuenta tanto la disposición de controles e indicadores en la pantalla como la forma en la que éstos muestran la información.

La disposición de controles e indicadores ha de ser adecuada para una correcta visualización de todos o al menos de aquellos que se considere necesario poder localizar en un primer vistazo. Debe también permitir la colocación en segundo plano de aquellos que no se necesiten observar continuamente.

Debe tenerse en cuenta también la influencia de la colocación de estos componentes desde el punto de vista del rendimiento del operador basándonos ya sea en experiencia personal, o estudios propios o de terceros. Esta disposición es especialmente importante en aquellos casos en los que el centro de control vaya a monitorizar más de un vehículo [18], ya que influye notablemente en el desarrollo de la labor del operador.

En el diseño del Centro de Control Versátil, hemos tenido en cuenta todos estos factores y hemos tratado de dotar a la GUI de la mayor flexibilidad posible permitiendo una gran cantidad de configuraciones distintas. Además se permite cambiar algunos factores de visualización de los indicadores en base a las necesidades de cada situación y se incorporan algunas funcionalidades interesantes, como la posibilidad de duplicar algunos de los controles de la ventana principal y situarlos en una ventana secundaria que puede superponerse o colocarse junto a la GUI.

A continuación detallaremos cada uno de los componentes principales de la GUI y explicaremos brevemente su función.

3.1.1 Barra de Menús

Se trata de una barra de menús convencional, muy similar a la que podríamos observar en cualquier programa basado en un sistema de ventanas.

Ofrece algunas opciones como la posibilidad de marcar las pestañas (ver sección "Navegación por Pestañas") para que al cambiar a una pestaña (global o de un vehículo) el programa sitúe en primer plano todas las pestañas relacionadas. Además permitirá otras opciones como pueden ser la de guardado de datos, introducción de marcadores en el mapa o acceso a gráficas entre otros.

También contará con un menú propio para los ítems que desplegarán los controladores integrados en el Centro de Control Versátil (ya sean para todos o sólo parte de los vehículos monitorizados).

La barra de menús se encuentra referenciada por el número 1, en color azul en la Figura 11.

3.1.2 Panel de Mapa

El panel de mapa es quizá la parte más importante de todo el interfaz. En este panel aparecerá un mapa de OpenStreetMaps [5] en el que se mostrarán los vehículos que se desee monitorizar.

Gracias al documento XML de configuración, el usuario podrá elegir el tipo de imagen representativa del vehículo que desea mostrar en el mapa (una imagen a la escala del mismo o el indicador por defecto, un triángulo con el color que representa a dicho vehículo). Esta imagen se moverá por el mapa al recibir las coordenadas (latitud, longitud y orientación) del vehículo.

Adicionalmente el mapa se puede configurar para su funcionamiento en dos modos: El primero y más habitual contando con conexión a internet, en cuyo caso cargará el mapa bajo demanda al movernos por el mismo. El segundo, fuera de línea, utilizando los mapas de la zona previamente descargados. En este caso deberemos indicarle al programa la ruta donde hayamos guardado los mapas en nuestro PC.

Estas dos posibilidades de configuración ofrecen una gran flexibilidad a la hora de realizar misiones en zonas remotas ya que, por una parte permite un ahorro considerable de energía en nuestros dispositivos al evitar una conexión y descarga continua de datos para el mapa y por otra, aún más importante, permite que el programa funcione correctamente en condiciones en las que quizá de otra forma no podría como misiones en zonas en las que no hay posibilidad de conexión.

El panel de mapa aparece referenciado con el número 2, en rojo, en la Figura 11.

3.1.3 Panel Vertical

El panel vertical está formado por dos subpaneles, superior e inferior. En uno de ellos se introducirá el subpanel de alarmas y en el otro el subpanel de selectores.

Estos dos subpaneles podrán mostrarse arriba o abajo según lo indique el usuario en el archivo de configuración XML permitiendo adecuar la visibilidad de estos elementos a sus necesidades o preferencias.

Al igual que en el panel horizontal, se mostrará una pestaña global para elementos comunes que se deseen visualizar en primer plano y una pestaña propia de cada vehículo monitorizado.

En el subpanel de alarmas se insertarán todas aquellas alarmas programadas por el usuario. Éstas sonaran y/o mostraran avisos al alcanzar los límites programados de la variable monitorizada correspondiente.

En el subpanel de selectores se insertarán todos los elementos tipo selector que permitirán desplegar ventanas adicionales que contengan cualquiera de los indicadores que podríamos mostrar de forma normal en otros paneles del interior del interfaz de usuario.

Ambos tipos de elemento se insertarán, al igual que el resto, en el orden dado por el archivo de configuración XML. Tampoco se ha establecido un número máximo de elementos ya que si el número de elementos a mostrar excede la parte visible del panel, aparecerá una barra de desplazamiento en la parte derecha de los subpaneles que nos permitirá navegar por los elementos que queden ocultos.

El panel vertical se encuentra referenciado por el número 3, en verde, en la Figura 11.

3.1.4 Panel Horizontal

El panel horizontal es el lugar donde se irán insertando los indicadores y controladores pertenecientes a los vehículos representados.

Contará con una pestaña global para los indicadores comunes más importantes que queramos poder ver al mismo tiempo a pesar de pertenecer a vehículos distintos y una pestaña para los indicadores de cada vehículo.

Los indicadores podrán colocarse en el orden que se desee, determinado por el archivo de configuración XML.

Gracias a estas dos posibilidades, contaremos con una gran cantidad de configuraciones posibles para este panel, que nos permitirá configurarlo según las preferencias del operador y las necesidades de la misión.

No existe límite a la hora de introducir indicadores en la barra horizontal, ya que en caso de no entrar todos en la parte visible de la ventana, ésta se alargará y permitirá navegar por ellos gracias a una barra de desplazamiento inferior.

El panel horizontal se encuentra referenciado por el número 4, en negro, en la Figura 11.

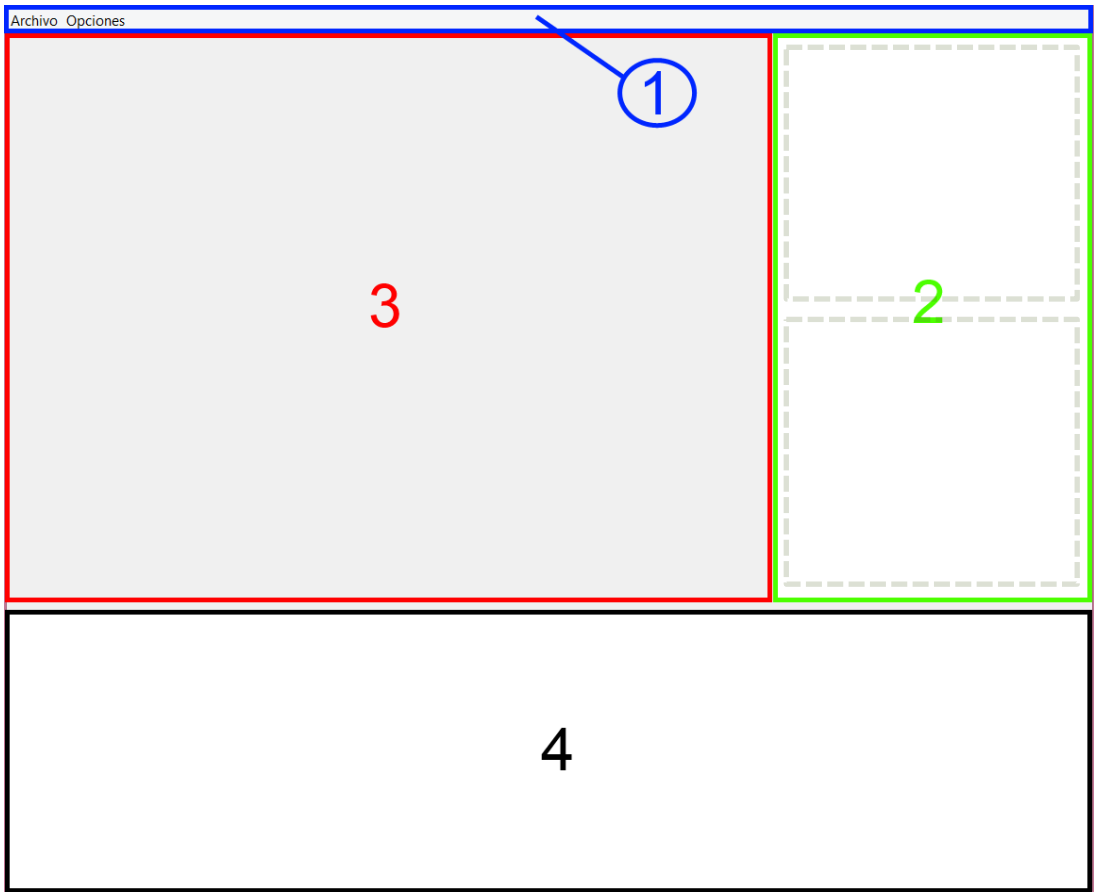

Figura 11: Estructura General de la GUI

3.2 Configuraciones Principales

A continuación explicaremos las cuatro configuraciones principales que se pueden dar alterando la disposición de los elementos principales anteriormente mencionados. No describiremos en detalle la disposición de los indicadores dentro de cada panel ni la posición (superior - inferior) de los subpaneles de alarmas y selectores dentro de la barra vertical ya que el número de posibilidades de configuración en base al orden de indicadores es muy alto.

Respecto a los elementos contenedores principales y el mapa, pueden colocarse en las siguientes posiciones:

- Mapa: arriba izquierda, arriba derecha, abajo izquierda y abajo derecha
- Barra vertical: arriba izquierda, arriba derecha, abajo izquierda y abajo derecha
- Barra horizontal: arriba o abajo

No todas las combinaciones de posiciones de los elementos contenedores son posibles. De hecho la configuración del interfaz de usuario quedará dispuesta en base a las limitaciones de los elementos que ya han sido colocados. Es decir, si por ejemplo se decide colocar el mapa arriba a la izquierda, la barra horizontal quedará limitada a la zona inferior y la barra vertical a la zona superior derecha ya que sería la única forma en que encajarían correctamente en el espacio habilitado al efecto.

El programa comprobará automáticamente si la configuración elegida es correcta o si por el contrario hay algún conflicto entre elementos, en cuyo caso se mostraría un mensaje de error al cargar el archivo de configuración XML.

3.2.1 Configuración A

Se trata de la configuración por defecto. Contará con el mapa en la zona superior izquierda, la barra horizontal en la zona inferior y la vertical en la zona superior derecha. Podemos observar un ejemplo de la disposición de los componentes gráficos en esta configuración en la Figura 12 (superior).

3.2.2 Configuración B

En esta configuración el mapa se situará en la zona superior a la derecha, la barra vertical en la zona superior izquierda y la barra horizontal en la zona inferior. Podemos observar un ejemplo de la disposición de los componentes gráficos en esta configuración en la Figura 12 (inferior).

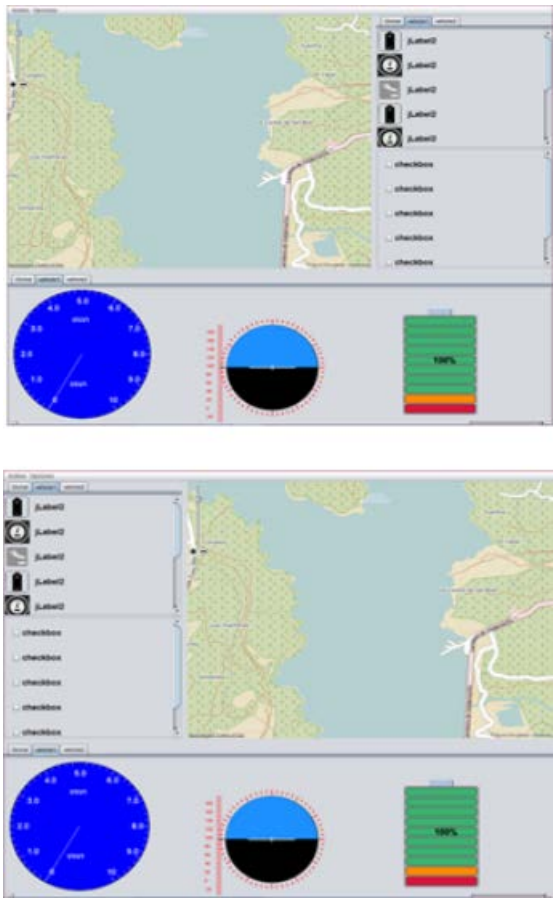

Figura 12: Configuración A (Superior) y Configuración B (Inferior)

3.2.3 Configuración C

En esta configuración el mapa se situará en la zona inferior izquierda, la barra horizontal en la zona superior y la barra vertical en la zona inferior derecha. Podemos observar un ejemplo de la disposición de los componentes gráficos en esta configuración en la Figura 13 (superior).

3.2.4 Configuración D

En esta configuración el mapa se situará en la zona inferior derecha, la barra horizontal en la zona superior y la barra vertical en la zona inferior izquierda. Podemos observar un ejemplo de la disposición de los componentes gráficos en esta configuración en la Figura 13 (inferior).

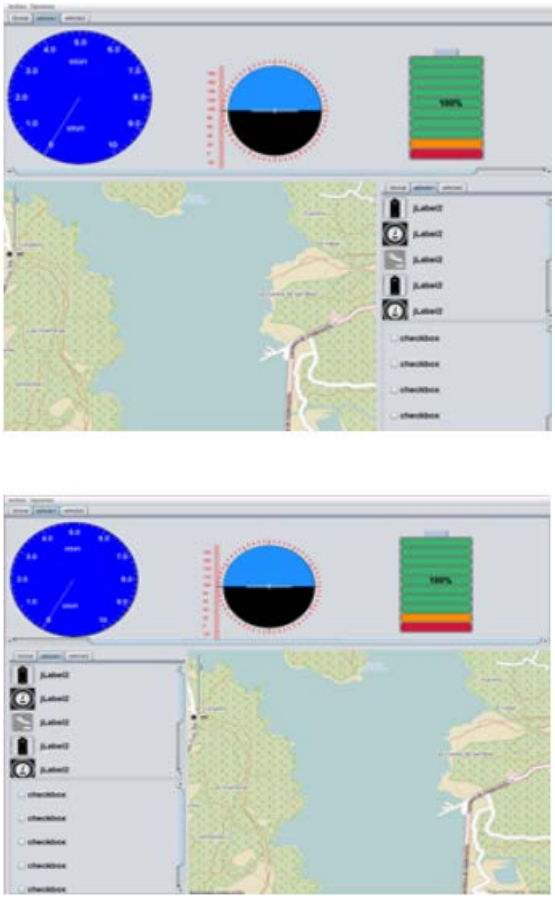

Figura 13: Configuración C (Superior) y Configuración D (Inferior)

3.3 Elementos de Control y Visualización

En esta sección analizaremos los indicadores creados para realizar la demo del Centro de Control Versátil.

Estos elementos se encargarán de mostrar los valores monitorizados del estado de los vehículos. Han sido diseñados para utilizarlos con cualquier tipo de vehículo autónomo. Esto dota al Centro de Control Versátil de una flexibilidad casi total a la hora de reconfigurar el interfaz de usuario ya que con unos sencillos valores introducidos en el archivo de configuración XML, los indicadores se reconfigurarán de forma totalmente transparente al usuario, lo que sería prácticamente impensable en otro tipo de centro de control.

Aunque en centros de control comerciales pueden encontrarse otros tipos de indicadores, para la demostración del prototipo se han creado sólo los más útiles o imprescindibles para demostrar la funcionalidad del Centro de Control Versátil.

3.3.1 Velocímetro

Se trata de un indicador que plasmará de forma gráfica la velocidad del vehículo.

Está formado por una esfera con una serie de valores marcados en intervalos y una aguja que indicará el valor actual de velocidad recibido. El fondo del velocímetro mostrará el color del vehículo al que representa y los valores estarán normalizados, de forma que el usuario pueda elegir el valor mínimo y el valor máximo mostrado en la esfera, como podemos observar en la Figura 14.

Además contará con dos etiquetas, una de ellas con el identificador asignado al vehículo al que representa y la otra con las unidades elegidas por el usuario para mostrar los valores de velocidad (p. ej. Km/h, nudos, mph).

Si el operador lo considera necesario, podría monitorizarse la velocidad de varios vehículos mostrando varias agujas de diferentes colores en la misma esfera.

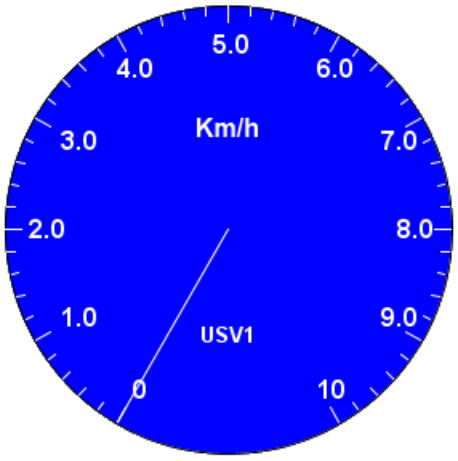

Figura 14: Velocímetro

3.3.2 Horizonte Artificial

Este indicador mostrará gráficamente los valores de cabeceo, alabeo y guiñada del vehículo (normalmente UAV) al que representa.

Se trata de una esfera en la que se representa el cielo y el suelo para ofrecer una referencia gráfica que, junto con una línea de status mostrará de forma descriptiva la orientación del vehículo.

Podemos ver el aspecto gráfico de este componente en la Figura 15.

El *cabeceo* (Pitch) se mostrará mediante la línea blanca central que actuará como referencia respecto al horizonte y se cuantificará con los valores representados en la escala situada a la izquierda.

El *alabeo* (Roll) se mostrará gráficamente inclinando la línea del horizonte artificial.

La *guiñada* (Yaw) se mostrará gracias a una brújula horizontal en la base del horizonte artificial.

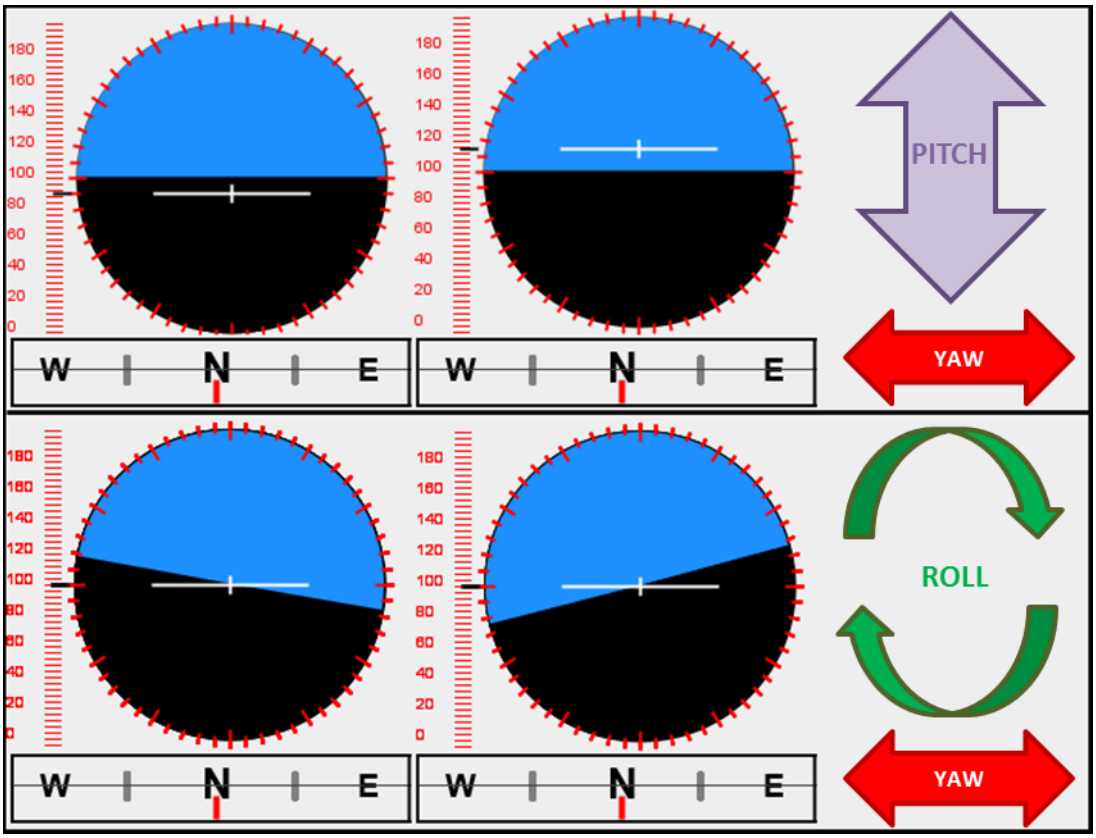

Figura 15: Horizonte Artificial

3.3.3 Grupo de Valores Numéricos

Se trata de un panel en el que se agruparán valores numéricos introducidos por el usuario en el documento XML. Cuenta con una barra de desplazamiento en la parte derecha, de tal forma que si se introducen más valores de los que pueden mostrarse en el cuadro que aparece en la pantalla principal, éste se desplazará y podremos navegar por él mostrando los que sean necesarios. También podrá introducirse más de un grupo de valores numéricos si así se desea. Podemos ver un grupo de valores numéricos en la Figura 16.

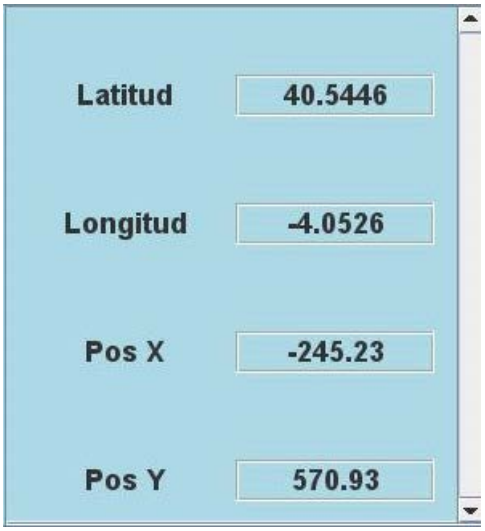

Figura 16: Grupo de Valores Numéricos

3.3.4 Valor Numérico

Los valores numéricos se representarán en un panel rectangular con dos etiquetas, una en la parte izquierda en la que figurará el texto identificativo que se elija para ese valor numérico y otra en la parte izquierda en la que se mostrará el valor numérico que se lea del vehículo. Podemos ver el detalle de uno de los valores numéricos integrados en el grupo de valores numéricos anterior en la Figura 17.

El formato de estos valores numéricos se puede configurar desde el documento XML en el que se elige el número de cifras decimales necesario para mostrarlas.

Figura 17: Valor Numérico

3.3.5 Alarma

Las alarmas se visualizarán simbólicamente en forma de panel en el que se mostrará una imagen y un texto, ambos orientativos y configurables desde el documento XML. Podemos ver el ejemplo de dos alarmas con una imagen que representa al tipo de alarma y un texto orientativo en la Figura 18.

Internamente, la alarma contará con un valor numérico configurado por el usuario que, al ser alcanzado, lanzará un panel informativo en primer plano del interfaz de usuario y un sonido que se podrá activar de forma opcional.

Figura 18: Alarmas

3.3.6 Selectores

Los paneles de selección representan a un applet de entre los disponibles para mostrar en el interfaz de usuario y constarán de un panel con una casilla de selectores y un texto identificativo, como podemos observar en la Figura 19.

El usuario podrá configurar tantos selectores como quiera e introducirlos en el interfaz, de forma que al pulsar sobre ellos se abrirá una ventana emergente que permitirá visualizar cualquier applet de forma independiente al interfaz de usuario principal. La ventana se ocultará si vuelve a pulsar sobre la casilla de selector pero seguirá activa para poder volver a mostrarla en cualquier momento.

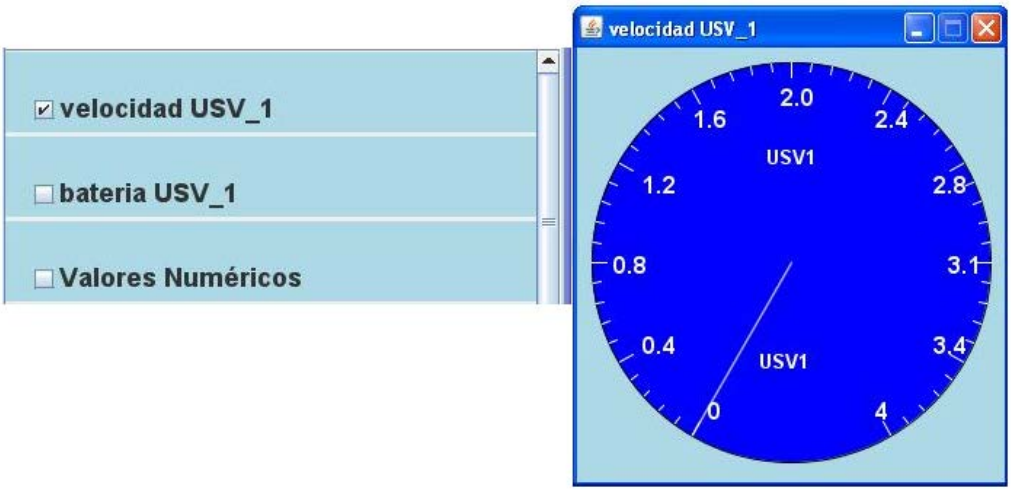

Figura 19: Selector Junto a su Ventana Emergente

3.3.7 Indicador de Batería

El applet indicador de batería mostrará gráficamente el porcentaje de batería que envíe el vehículo. Contará además con un indicador numérico para una mayor precisión. Podemos ver un ejemplo del indicador de carga de batería con dos porcentajes distintos tanto en valor numérico como representado gráficamente en la Figura 20.

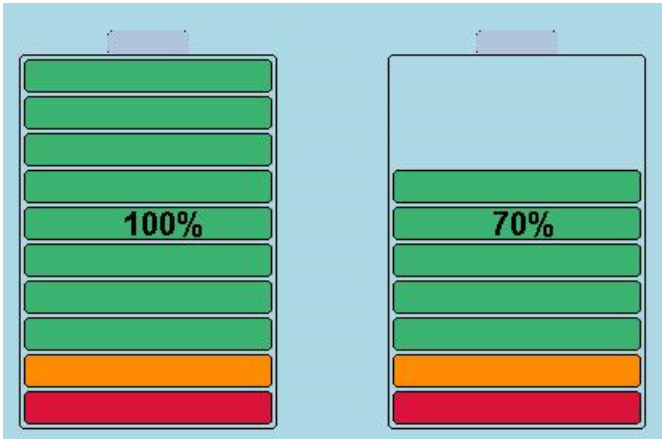

Figura 20: Indicador de Carga de Batería

3.3.8 Gráficas

Se ha añadido una herramienta basada en JFreeChart [22] que crea una ventana emergente en la que se visualizará una sencilla gráfica XY que mostrará el comportamiento de una de las variables monitorizadas por el Centro de Control respecto al tiempo.

Permitirá configurar al usuario el nombre de la gráfica, el nombre a mostrar en el eje X y el nombre a mostrar en el eje Y, además de la variable a mostrar. También se podrá introducir cada cuánto tiempo se desea capturar la variable monitorizada y el número de valores a mostrar (p.ej. capturar la velocidad cada 10 segundos y mostrar 10 valores de la misma). Podemos ver un ejemplo Ilustrativo en la Figura 21, en la que se muestran 10 valores de la velocidad de un USV capturados en intervalos de 1 segundo.

Figura 21: Gráfica Velocidad - Tiempo

3.4 Funcionalidades Adicionales

El interfaz de usuario cuenta además con algunas funcionalidades adicionales incorporadas en zonas clave con la intención de potenciar las posibilidades de personalización y con ello mejorar la ergonomía del interfaz de usuario.

3.4.1 Navegación por las Pestañas

Como se ha explicado en secciones anteriores, el interfaz de usuario cuenta con una división en la que cada vehículo es asignado a una pestaña identificada con el nombre del vehículo y un color de forma que alternando las pestañas se visualizarán los indicadores y controladores de cada uno de ellos (además de la pestaña global en la que se pueden mostrar simultáneamente indicadores de varios vehículos).

Inicialmente el programa estaba configurado sin bloqueo de pestañas, es decir, podía visualizarse simultáneamente la pestaña de un vehículo en la barra horizontal y la pestaña de otro vehículo en la barra vertical.

Sin embargo, los ensayos y pruebas de funcionamiento realizados con el Centro de Control Versátil se llegó a la conclusión de que podría resultar confusa una disposición de pestañas en la que no coincidiera el vehículo en ambas zonas del interfaz, pudiendo alterar la percepción del operador. Aun así, también puede resultar útil (o necesario) visualizar pestañas de distintos vehículos dependiendo de las preferencias del usuario.

Por lo tanto, al parecer razonables ambas configuraciones de navegación, ambas se han incluido en el interfaz. Por defecto se ha configurado la opción de bloqueo de pestañas en la que al seleccionar la pestaña de uno de los vehículos en la barra horizontal o vertical, automáticamente se situará en primer plano la pestaña correspondiente a ese vehículo en la otra barra. Si se desea desbloquear la navegación de pestañas tan solo se ha de acudir a la barra de menú y pulsar en **Opciones/Bloqueo de pestañas**.

3.4.2 Duplicado y Extracción de Pestaña

Otra posibilidad de configuración incluida en el Centro de Control Versátil es la de realizar el duplicado de una de las pestañas (ya sea de la barra horizontal o vertical) y extraerla a una ventana emergente en la que se podrá visualizar de forma independiente al interfaz principal.

Para realizar el duplicado y extracción se deberá hacer click derecho sobre la pestaña que se quiera extraer. El programa solicitará confirmación para realizar la extracción y al aceptarla aparecerá un duplicado de la pestaña con los controles e indicadores necesarios pertenecientes a ese vehículo.

3.4.3 Guardado de Datos

Durante el desarrollo de misiones con vehículos no tripulados suele resultar necesario guardar datos referentes a los vehículos, su posición y el status de sus componentes en cada instante de tiempo, de forma que los usuarios puedan analizarlos posteriormente.

Como una primera aproximación a este almacenamiento de datos para nuestro prototipo, acudiendo a la barra de menú y pulsando sobre **Archivo/Guardar Datos**, el Centro de Control Versátil abrirá una ventana que permitirá seleccionar la ubicación y el nombre de un fichero tipo TXT que contendrá los datos desde el inicio de la conexión hasta la finalización de la misma para cada uno de los vehículos. Más adelante, como trabajo futuro, se planteará el estudio de otros formatos para el guardado de datos, así como su persistencia (base de datos).

Cada variable se almacenará en una línea con un formato sencillo que incluirá el identificador del vehículo, el identificador de la variable y su valor.

3.4.4 Introducción de Marcadores

Durante los ensayos del centro de control, se observó que resultaría muy interesante poder introducir hitos o marcadores en el mapa para señalizar ciertos puntos importantes en el desarrollo de la misión (p. ej. punto de inicio, punto final, objetivos, boyas, bolardos, obstáculo) de forma que podamos visualizar de forma gráfica la posición relativa de los vehículos respecto a esos puntos.

Acudiendo a la barra de menú y pulsando **Opciones/Introducir Marcador** aparecerá una ventana en la que se podrá introducir las coordenadas (Longitud y Latitud) y un identificador para el marcador, que aparecerá inmediatamente sobre el mapa al pulsar **Aceptar**.

Se agregará en un futuro la posibilidad de introducir marcadores individuales y grupos de marcadores que delimiten zonas o marquen fronteras mediante un documento XML y la posibilidad de agregar una alarma de proximidad a las coordenadas que éstos señalicen.

3.4.5 Mostrar Ruta

Durante el desarrollo de una misión o en los ensayos de movimiento programado de alguno de los vehículos puede resultar muy útil poder visualizar sobre el mapa la trayectoria que éste ha seguido sobre el terreno.

Por esto, en el Centro de Control Versátil se ha introducido la posibilidad de mostrar y ocultar la ruta acudiendo a la barra de menú y marcando/desmarcando el selector en **Opciones/Mostrar Ruta**.

También se ofrece la posibilidad de borrar la trayectoria almacenada hasta el momento y comenzar a guardar una nueva pulsando **Opciones/Resetear Ruta**.

CAPÍTULO 4: DISEÑO ESTRUCTURA DE CONFIGURACIÓN DINÁMICA

En este capítulo analizaremos el método utilizado para que un usuario configure el Centro de Control Versátil en base a sus necesidades o preferencias.

Se darán las pautas para elaborar el fichero de configuración XML, describiendo los elementos a utilizar y su forma correcta de uso. Además, se expondrá brevemente la estructura utilizada para la lectura del fichero así como el método utilizado para comprobar su corrección.

4.1 ¿Por Qué Formato XML?

El formato XML (eXtensible Markup Language) es un lenguaje de marcado standard desarrollado por W3C (World Wide Web Consortium) [23] utilizado para almacenar datos de forma legible. Se propone como un standard para el intercambio de información estructurada entre diferentes plataformas y se puede utilizar en bases de datos, editores de texto, hojas de cálculo, etc.

Se ha elegido este formato a pesar de tener algunas desventajas como el complejo mapeo del árbol básico de XML hacia algunos lenguajes de programación a la hora de intercambiar datos altamente estructurados (dificultad que ha intentado minimizarse en este proyecto simplificando al máximo las estructuras de datos de modo que el árbol obtenido sea lo más sencillo posible y minimizando la complejidad en el algoritmo de lectura del mismo) ya que cuenta con algunas notables ventajas como:

- Su extensibilidad, que permite añadir nuevas etiquetas en cualquier momento, y por lo tanto agregar al centro de control indicadores o controles de forma sencilla.
- El analizador es standard, lo que permite emplear cualquier tipo de analizador disponible en cualquier plataforma (Linux, Windows etc.) y con cualquier tipo de lenguaje de programación o base de datos. Esto facilita la estandarización de nuestra aplicación ya que permite comunicarse con una amplia gama de software de control de vehículos autónomos independientemente del lenguaje y plataforma en el que hayan sido diseñados.
- Añadir a los datos un significado y asociarles un contexto, lo que permite una gran flexibilidad a la hora de tratarlos en el centro de control y prepararlos para una visualización lo más correcta e intuitiva posible por parte del operador.

4.2 Documento de Configuración XML

Una vez explicadas las ventajas y desventajas con las que cuenta un documento XML, explicaremos la estructura que hemos decidido para el fichero de configuración del Centro de Control Versátil.

Los documentos en formato XML cuentan con una herramienta llamada DTD (Document Type Definition, Definición de Tipo de Documento) [24] [25] en la que se describe la estructura que debe tener, elementos con los que puede contar, orden y número de elementos que se pueden incluir. Esta DTD se puede ampliar cuando sea necesario añadir algún tipo de componente o variable al centro de control.

4.2.1 Descripción de Elementos

- **TÍTULO**

Elemento que representará el nombre o título que se le quiera poner a la ventana principal del Centro de Control Versátil.

<!ELEMENT title (#PCDATA)>

Ejemplo:

<title>Centro de Control Versátil: USV + UAV</title>

- **VEHÍCULO**

Deberá contener el tipo de vehículo y los elementos que se usan para mostrarlo en la unidad de control: su identificador y la imagen que se muestra en el mapa (si no se incluye se mostrará un puntero en forma de triángulo).

<!ELEMENT vehicle (id,(img_path)?)>
<!ATTLIST vehicle type (UAV | USV | UGV) #REQUIRED>
<!ELEMENT id (#PCDATA)>
<!ELEMENT img_path (#PCDATA)>

Ejemplo:

```
<vehicle type="USV">
        <id>identificador</id>
        <img_path>C:\usuario\desktop\imagenes_usv</img_path>
</vehicle>
```

- **PUERTOS**

En los casos en los que sea necesario, contendrá el puerto de entrada y de salida en los que se hará la conexión respecto a la unidad de control.

<!ELEMENT port (in_port, out_port)>
<!ELEMENT in_port (#PCDATA)>

<!ELEMENT out_port (#PCDATA)>

Ejemplo:
```
<port>
        <in_port>1234</in_port>
        <out_port>5678</out_port>
</port>
```

- **CONTROLADOR**

En los casos en los que desee incorporar un controlador al Centro de Control Versátil, se deberá dar la ruta del ejecutable del mismo para que el configurador lo localice y lo invoque al pulsar el botón asignado.

<!ELEMENT controller (id, path)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT path (#PCDATA)>

Ejemplo:
```
<controller>
        <id>Controlador USV1</id>
        <path>c:\controladores\controlador_usv1.jar</path>
</controller>
```

- **MAPA**

Representa al panel que contendrá el mapa y en el que se mostrarán los vehículos.

Se trata de uno de los elementos principales del centro de control y su inclusión será obligatoria.

Su función principal será marcar la ubicación del mapa en el interfaz, alrededor del cual se agruparán el resto de elementos.

Llevará por defecto la posición top-left (arriba-izquierda) en caso de que no se asigne una específicamente.

<!ELEMENT map EMPTY>
<!ATTLIST map position (top-left | top-right | bottom-left | bottom-right) top-left>

Ejemplo:
```
<map position="top-right" />
```

- **BARRA HORIZONTAL**

Barra horizontal con desplazamiento que contendrá indicadores y demás objetos necesarios. Se dividirá en pestañas en las que se mostrará una pestaña por vehículo y una pestaña global en la que se mostrarán los indicadores más críticos elegidos por el usuario marcando el elemento en cuestión con el atributo "global=yes". La barra señalizará su posición en el interfaz por medio del atributo position.

Los indicadores y controles se mostrarán de izquierda a derecha en la barra horizontal en el orden en que han sido incluidos en el elemento “horizontal bar” del documento XML.

<!ELEMENT horizontal_bar (speedometer | artificialHorizon | battery | numericValueGroup)+>
<!ATTLIST horizontal_bar position (top | bottom) top>

Ejemplo:

```
<horizontal_bar>
        <battery global="no" >
                …
        </battery>
        <speedometer global="yes" >
                …
        </speedometer>
                .
                .
                .
        <artificialHorizon global="no" >
                …
        </artificialHorizon>
</horizontal_bar>
```

- **BARRA VERTICAL**

Barra vertical, dividida en dos partes. Una de las partes mostrará las alarmas mientras que la otra mostrará una serie de selectores con elementos que se visualizarán en ventanas emergentes de forma opcional bajo demanda del usuario.

La barra señalizará su posición en el interfaz por medio del atributo position. Se señalizará el orden del panel de alarmas y selectores mediante el atributo “position” de los mismos.

<!ELEMENT vertical_bar (alarm_bar, checkbox_bar)>
<!ATTLIST vertical_bar position (left | right) right>

Ejemplo:

```
<vertical_bar position="right">
        <alarm_bar position=”top”>
                <alarm name="battery" />
                        .
                        .
                        .
                <alarm name="battery" />
        </alarm_bar>
        <checkbox_bar position=”bottom”>
                <checkbox name="chart"/>
```

```
                .
                .
                .
            <checkbox name="chart"/>
        </checkbox_bar>
</vertical_bar>
```

- **BARRA DE ALARMAS**

Barra en la que se mostrarán las alarmas, especificará su posición en la barra vertical mediante el atributo position.

<!ELEMENT alarm_bar (alarm)+ >
<!ATTLIST alarm_bar position (top | bottom) top >

Ejemplo:

```
<alarm_bar position="top">
        <alarm name="battery" />
                .
                .
                .
        <alarm name="battery" />
</alarm_bar>
```

- **BARRA DE SELECTORES**

Barra en la que se visualizarán los selectores con elementos que se mostrarán en ventanas emergentes. Especificará su posición en la barra vertical mediante el atributo position.

<!ELEMENT checkbox_bar (checkbox)+ >
<!ATTLIST checkbox_bar position (top | bottom) bottom >

Ejemplo:

```
< checkbox _bar position="bottom">
        <checkbox name="speedometer"/>
                .
                .
                .
        <checkbox name="chart"/>
</checkbox_bar>
```

- **ALARMA**

Elemento que representará gráficamente una alarma asociada a una variable que se reciba en el centro de control y cuyo valor deberá fijarse por parte del usuario.

Cuando la variable supere el valor introducido. Se indicará mediante el atributo "type" si ha de ser mayor (greaterThan) o menor (lowerThan) que el valor marcado y se mostrará un mensaje por pantalla y/o un sonido que avise al operador.

La alarma podrá mostrar un icono descriptivo y un texto que se indicará al operador a qué variable representa la alarma y otro texto que aparecerá en el panel que se mostrará cuando salte la alarma. Adicionalmente se podrá elegir para la alarma uno de los siguientes colores: verde, ámbar o rojo.

Como en todos los indicadores o applets, contará con un identificador que permite conocer la variable a la que se encuentra asociada. Además habrá que indicar al Centro de Control Versátil el formato y cantidad de datos que le llegarán para tratarlos y representarlos correctamente. De esta forma se dota al centro de control de mucha flexibilidad, ya que se permite la recepción de datos de tipos muy diversos.

<!ELEMENT alarm (id,value,text,message,format)>
<!ATTLIST alarm name (battery | altitude | ... | speed) #REQUIRED >
<!ATTLIST alarm global (yes | no) #REQUIRED >
<!ATTLIST alarm type (greaterThan | lowerThan) #REQUIRED >
<!ATTLIST alarm sound (yes | no) no >
<!ELEMENT id (#PCDATA) >
<!ELEMENT value (#PCDATA) >
<!ATTLIST value color (green | yellow | red) #IMPLIED >
<!ATTLIST value type (greaterThan | lowerThan) >
<!ELEMENT text (#PCDATA)>
<!ELEMENT message (#PCDATA)>
<!ELEMENT format (#PCDATA)>

Ejemplo:

```
<alarm name="battery"  global="no" sound="yes">

        <id >alarma1</id>
        <value  color="yellow" type=”greaterThan”> 50 </value>
        <text>Batería Medio</text>
        <message>Precaución, la batería ha llegado al 50% de carga</message>
        <format>double,1</format>

</alarm>
```

- **HORIZONTE ARTIFICIAL**

Elemento que mostrará gráficamente cabeceo, alabeo y guiñada de un vehículo.

<!ELEMENT artificialHorizon (id,format)>
<!ATTLIST alarm global (yes | no) #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT format (#PCDATA) >

Ejemplo:

```
<artificialHorizon  global="no" >
```

```
        <id >HorizonteArtificial 1</id>
        <format>double,3</format>

</artificialHorizon>
```

- **CARGA DE BATERÍA**

Elemento que mostrará gráficamente el valor de la carga de batería del vehículo.

<!ELEMENT battery (id,format)>
<!ATTLIST alarm global (yes | no) #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT format (#PCDATA) >

Ejemplo:

```
<battery  global="no" >

        <id >batería 1</id>
        <format>integer,1</format>

</battery>
```

- **GRÁFICA**

Elemento que mostrará gráficamente el valor de la carga de batería del vehículo.

<!ELEMENT chart (id,xaxis,yaxis,interval,numElements,format)>
<!ATTLIST chart global (yes | no) #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT xaxis (#PCDATA) >
<!ELEMENT yaxis (#PCDATA) >
<!ELEMENT interval (#PCDATA) >
<!ELEMENT numElements (#PCDATA) >
<!ELEMENT format (#PCDATA) >

Ejemplo:

```
<chart  global="no" >

        <id >velocidad_tiempo</id>
        <xaxis>tiempo(s)</xaxis>
        <yaxis>velocidad(nudos)</yaxis>
        <interval>10</interval>
        <numElements>10</numElements>
        <format>double,1</format>

</chart>
```

- **SELECTOR**

Elemento que mostrará por medio de un selector y un texto, indicadores, controles o alarmas que están siendo ejecutados en segundo plano y por tanto no se muestran en paneles principales (o se muestran pero se desea tener un duplicado externo por si se cambia de pestaña).

Cuando el usuario seleccione el selector, se mostrará una ventana emergente con el elemento al que representa. Se ocultará dicho elemento al desmarcar el selector.

<!ELEMENT checkbox (id,text,format)>
<!ATTLIST checkbox global (yes | no) #REQUIRED >
<!ATTLIST checkbox name (speedometer | numericValueGroup | alarm | artificialHorizon | ... | battery) #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT text (#PCDATA) >
<!ELEMENT format (#PCDATA) >

Ejemplo:

```
<checkbox name="speedometer" global=”no” name=”speedometer”>

    <id>vel1</id>
    <text>Velocímetro del USV#1</text>
    <format>double,1</format>

</checkbox>
```

- **VALOR NUMÉRICO**

Elemento que mostrará un valor numérico variable junto con un texto introducido por el usuario para identificarlo. Se podrá indicar la precisión (número de decimales).

Se definirá dentro de un grupo de valores numéricos.

<!ELEMENT numericValue (id, name,precision,format)>
<!ELEMENT id (#PCDATA) >
<!ELEMENT name (#PCDATA) >
<!ELEMENT precision (#PCDATA) >
<!ELEMENT format (#PCDATA) >

Ejemplo:

```
<numericValue >

    <id>num_alt1</id>
    <name>Indicador de altitud</name>
    <precision>2</precision>
    <format>double,1</format>
```

</numericValue>

- **GRUPO DE VALORES NUMÉRICOS**

Elemento que agrupará varios valores numéricos definidos por el usuario.

<!ELEMENT numericValueGroup (id,(numericValue)+)>
<!ATTLIST numericValueGroup global (yes | no) #REQUIRED >

Ejemplo:

```
<numericValueGroup   global="yes">

        <id>numValGroup1</id>
        <numericValue >
                …
        </numericValue>
                .
                .
                .
        <numericValue >
                …
        </numericValue>

 </numericValueGroup>
```

- **VELOCÍMETRO**

Elemento que mostrará la velocidad del vehículo al que representa.

Debido a la multitud de unidades e intervalos numéricos en que se puede representar la velocidad (nudos, mph, km/h…) el indicador se mostrará en el centro de Control Versátil en función los valores introducidos en ciertas etiquetas de este elemento. Mediante las etiquetas minSpeed y maxSpeed (mínima y máxima velocidad) se establecerá el intervalo de valores que se mostrarán en el velocímetro (p. ej. de 0 a 100), mediante la etiqueta unit (unidad) se mostrará en pantalla una etiqueta que indique la unidad de medida (p. ej. km/h).

El valor de la variable que recibe el velocímetro se normalizará para ser mostrado correctamente en la esfera que lo representa, y entre los valores mínimo y máximo introducidos.

<!ELEMENT speedometer (id,text,unit,minSpeed,maxSpeed,format)>
<!ATTLIST speedometer global (yes | no) #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT text (#PCDATA) >
<!ELEMENT unit (#PCDATA) >
<!ELEMENT minSpeed (#PCDATA) >
<!ELEMENT maxSpeed (#PCDATA) >
<!ELEMENT format (#PCDATA) >

Ejemplo:

```
<speedometer  global="yes" >

        <id>vel1</id>
        <text>UAV_1</text>
        <unit>Km/h</unit>
        <minSpeed>0</minSpeed>
        <maxSpeed>100</maxSpeed>
        <format>double,1</format>

</speedometer>
```

4.2.2 Definición DTD

En la especificación de la DTD se indicará la estructura del documento XML así como de cada uno de sus elementos.

```
<? xml version="1.0" standalone"yes">
<!DOCTYPE config [

<! ELEMENT config (title, vehicle, port, controller?, map, chart*, horizontal_bar, vertical_bar)>

<!-- Elemento Título -->
<!ELEMENT  title (#PCDATA)>

<!-- Elemento Vehículo -->
<!ELEMENT  vehicle (id, (img_path)?)>
<!ATTLIST vehicle  type  (UAV | USV | UGV ) #REQUIRED>
<!ELEMENT  id  (#PCDATA)>
<!ELEMENT  img_path  (#PCDATA)>

<!-- Elemento Puerto -->
<!ELEMENT port (in_port, out_port)>
<!ELEMENT in_port  (#PCDATA)>
<!ELEMENT out_port  (#PCDATA)>

<!-- Elemento Controlador -->
<!ELEMENT controller (id, path)>
<!ELEMENT id  (#PCDATA)>
<!ELEMENT path  (#PCDATA)>

<!-- Elemento Mapa -->
<!ELEMENT map EMPTY>
<!ATTLIST map position (top-left | top-right | bottom-left | bottom-right) top-left>
```

```
<!-- Elemento Gráfica-->
<!ELEMENT chart (id,xaxis,yaxis,interval,numElements,format)>
<!ATTLIST  chart  global  (yes | no) #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT xaxis (#PCDATA) >
<!ELEMENT yaxis (#PCDATA) >
<!ELEMENT interval (#PCDATA) >
<!ELEMENT numElements (#PCDATA) >
<!ELEMENT format (#PCDATA) >

<!-- Elemento Barra Horizontal -->
<!ELEMENT horizontal_bar  (speedometer | artificialHorizon | battery |
numericValueGroup)+>
<!ATTLIST  horizontal_bar  position (top | bottom) top>

<!-- Elemento Barra Vertical -->
<!ELEMENT vertical_bar  (alarm_bar, checkbox_bar)>
<!ATTLIST  vertical_bar  position (left | right) right>

<!-- Elemento Barra de Alarmas -->
<!ELEMENT alarm_bar  (alarm)+ >
<!ATTLIST  alarm_bar  position (top | bottom) top >

<!-- Elemento Alarma -->
<!ELEMENT alarm (id, value, text, message, format)>
<!ATTLIST  alarm  name  (battery | altitude | ... | speed)  #REQUIRED >
<!ATTLIST  alarm  global  (yes | no) #REQUIRED >
<!ATTLIST  alarm  type  (greaterThan | lowerThan)  #REQUIRED >
<!ATTLIST alarm sound (yes | no) no >
<!ELEMENT id (#PCDATA) >
<!ELEMENT value (#PCDATA) >
<!ATTLIST  value  color  (green | yellow | red)  #IMPLIED >
<!ELEMENT text (#PCDATA)>
<!ELEMENT message (#PCDATA)>
<!ELEMENT format (#PCDATA)>

<!-- Elemento Barra Checkbox -->
<!ELEMENT checkbox_bar (checkbox)+ >
<!ATTLIST  checkbox_bar  position (top | bottom) bottom >

<!-- Elemento Checkbox -->
<!ELEMENT checkbox  (id, text, format)>
<!ATTLIST  checkbox global (yes | no) #REQUIRED >
<!ATTLIST  checkbox name  (speedometer | numericValueGroup | alarm |
artificialHorizon | ... | battery )  #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT  text (#PCDATA) >
```

```
<!ELEMENT format (#PCDATA)>

<!-- Elemento Horizonte Artificial -->
<!ELEMENT artificialHorizon (id, format)>
<!ATTLIST  artificialHorizon  global  (yes | no) #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT format (#PCDATA)>

<!-- Elemento Batería -->
<!ELEMENT battery (id, format)>
<!ATTLIST  battery  global  (yes | no) #REQUIRED >
<!ELEMENT id (#PCDATA) >
<!ELEMENT format (#PCDATA)>

<!-- Elemento Valor Numérico -->
<!ELEMENT numericValue  (id, name, precision, format)>
<!ELEMENT  id (#PCDATA) >
<!ELEMENT  name (#PCDATA) >
<!ELEMENT  precision (#PCDATA) >
<!ELEMENT format (#PCDATA)>

<!-- Elemento Grupo de Valores Numéricos -->
<!ELEMENT numericValueGroup  (id, (numericValue)+)>
<!ATTLIST  numericValueGroup  global  (yes | no) #REQUIRED >

<!-- Elemento Velocímetro -->
<!ELEMENT speedometer  (id, text, unit, minSpeed, maxSpeed, format)>
<!ATTLIST  speedometer  global  (yes | no) #REQUIRED >
<!ELEMENT  id (#PCDATA) >
<!ELEMENT  text (#PCDATA) >
<!ELEMENT  unit (#PCDATA) >
<!ELEMENT  minSpeed (#PCDATA) >
<!ELEMENT  maxSpeed (#PCDATA) >
<!ELEMENT format (#PCDATA)>

]>
```

4.2.3 Ejemplo de Documento de Configuración XML

A continuación mostraremos un ejemplo de archivo de configuración XML totalmente funcional utilizado para cargar los elementos de un vehículo en el Centro de Control Versátil.

Después del contenido del archivo XML se introducirá una captura de pantalla del interfaz gráfico de usuario resultante (Figura 22).

```
<config>

    <title>Centro de Control Versátil</title>

    <!-- Elemento Vehículo -->
    <vehicle type="USV">
        <id>USV1</id>
        <img_path>C:\\Desarrollo\\GCS\\img</img_path>
    </vehicle>

    <!-- Elemento Puerto -->
    <port>
        <in_port>2233</in_port>
        <out_port>4321</out_port>
    </port>

    <!-- Elemento Mapa -->
    <map position="top-left"/>

    <!-- Elemento Barra Horizontal -->
    <horizontal_bar position="bottom">
        <!-- Elemento Horizonte Artificial -->
        <artificialHorizon global="no">
            <id>ah1</id>
            <format>double,3</format>
        </artificialHorizon>

        <!-- Elemento Bateria -->
        <battery global="yes">
            <id>b1</id>
            <format>integer,1</format>
        </battery>
```

```
<!-- Elemento Grupo de Valores Numericos -->
<numericValueGroup global="yes">
        <id>nvg1</id>

        <numericValue>
                <id>lat</id>
                <name>Latitud</name>
                <precision>4</precision>
                <format>double,1</format>
        </numericValue>

        <numericValue>
                <id>long</id>
                <name>Longitud</name>
                <precision>4</precision>
                <format>double,1</format>
        </numericValue>

        <numericValue>
                <id>posx</id>
                <name>Pos X</name>
                <precision>2</precision>
                <format>double,1</format>
        </numericValue>

        <numericValue>
                <id>posy</id>
                <name>Pos Y</name>
                <precision>2</precision>
                <format>double,1</format>
        </numericValue>

</numericValueGroup>

<!-- Elemento Velocimetro -->
<speedometer global="yes">
        <id>s1</id>
        <text>USV1</text>
```

```
                    <unit>Km/h</unit>
                    <minSpeed>0</minSpeed>
                    <maxSpeed>4</maxSpeed>
                    <format>double,1</format>
                </speedometer>

        </horizontal_bar>

        <!-- Elemento Barra Vertical -->
        <vertical_bar position="right">

                <alarm_bar position="top">
                    <!-- Elemento Alarma -->
                    <alarm name="speed" global="yes">
                        <id>alarm1</id>
                        <value color="red" type="greaterThan">2.5</value>
                        <text>Velocidad > 2.5</text>
                        <message>Precaucion, Velocidad Alta!</message>
                        <format>double,1</format>
                    </alarm>
                    <alarm name="battery" global="yes">
                        <id>alarm2</id>
                        <value color="yellow"
type="greaterThan">40</value>
                        <text>Batería al 40%</text>
                        <message>Precaucion, bateria al 40%</message>
                        <format>double,1</format>
                    </alarm>
                </alarm_bar>

                <checkbox_bar position="bottom">
                    <checkbox global="yes" name="speedometer" >
                        <id>check1</id>
                        <text>velocidad USV_1</text>
                        <format>double,1</format>
                    </checkbox>
                    <checkbox global="yes" name="battery" >
                        <id>check2</id>
                        <text>bateria USV_1</text>
```

```
                    <format>double,1</format>
                </checkbox>
                <checkbox global="yes" name="numericValueGroup" >
                    <id>check3</id>
                    <text>Valores Numéricos</text>
                    <format>double,1</format>
                </checkbox>
            </checkbox_bar>

        </vertical_bar>

</config>
```

Figura 22: Centro de Control Versátil Generado con Ejemplo XML

4.3 Lectura, Corrección y Manipulación de Datos XML

Una vez establecidas las bases de cómo debemos estructurar el documento XML de configuración para el Centro de Control Versátil, analizaremos el método de extracción de datos (o información) del documento, cómo se comprueba su corrección y cómo se manipulan éstos datos para mapearlos a los elementos del interfaz de usuario del centro de control.

4.3.1 Lectura del Documento XML

Para la lectura del documento XML en Centro de Control Versátil se ha elegido la API (Application Programming Interface) proporcionada por JDOM [20].

Se trata de un proyecto de código abierto cuyo objetivo es proporcionar una solución para lectura/escritura y manipulación de documentos XML. Enlazando las librerías que se proporcionan a nuestro proyecto, tenemos acceso a las clases y métodos que podamos necesitar para realizar las operaciones básicas con un documento XML.

4.3.2 Comprobación de Corrección de la DTD

A pesar de la sencillez de uso de JDOM, tiene una pequeña desventaja que se ha tratado de superar en este proyecto y es que hasta hoy y aunque hay en marcha proyectos para comprobar la corrección de una DTD, todavía no hay disponible ninguna clase o método que lo haga en el proyecto JDOM.

Dado el objetivo del Centro de Control Versátil, permitir a usuarios sin grandes conocimientos en programación modificar un centro de control en base a sus preferencias o necesidades, se ha considerado primordial la inclusión de un elemento que se encargue de comprobar si el documento de configuración XML introducido por el usuario tiene una estructura correcta. Esta comprobación puede resultar complicada ya que, como se ha comentado anteriormente, el árbol básico de elementos resultante puede ser bastante complejo.

Esta comprobación se ha dividido en dos bloques básicos. Por un lado debe realizarse sobre los Elementos, en los que hay que comprobar tanto la cantidad como el orden de elementos (y recursivamente sub-elementos de su interior) como sobre los datos que contienen. Por otro lado debe hacerse sobre los Atributos en los que se debe comprobar principalmente la cantidad y contenido de los mismos. Esto se logra mediante un método que realiza una lectura previa del fichero XML y se encarga de invocar para cada Elemento entrante a los métodos que comprueban la estructura del propio elemento y sub-elementos y la estructura de atributos del mismo.

En caso de encontrar algún elemento o atributo con estructura o contenido incorrecto, el método principal indicará mediante una ventana emergente (podemos ver un ejemplo en el que se ha introducido a propósito un error en un elemento de carga de batería en la Figura 23) al usuario que el fichero de configuración es incorrecto y en qué Elemento se encuentra la anomalía (para más detalles consultar Apéndice sobre JDOM y comprobación de DTD).

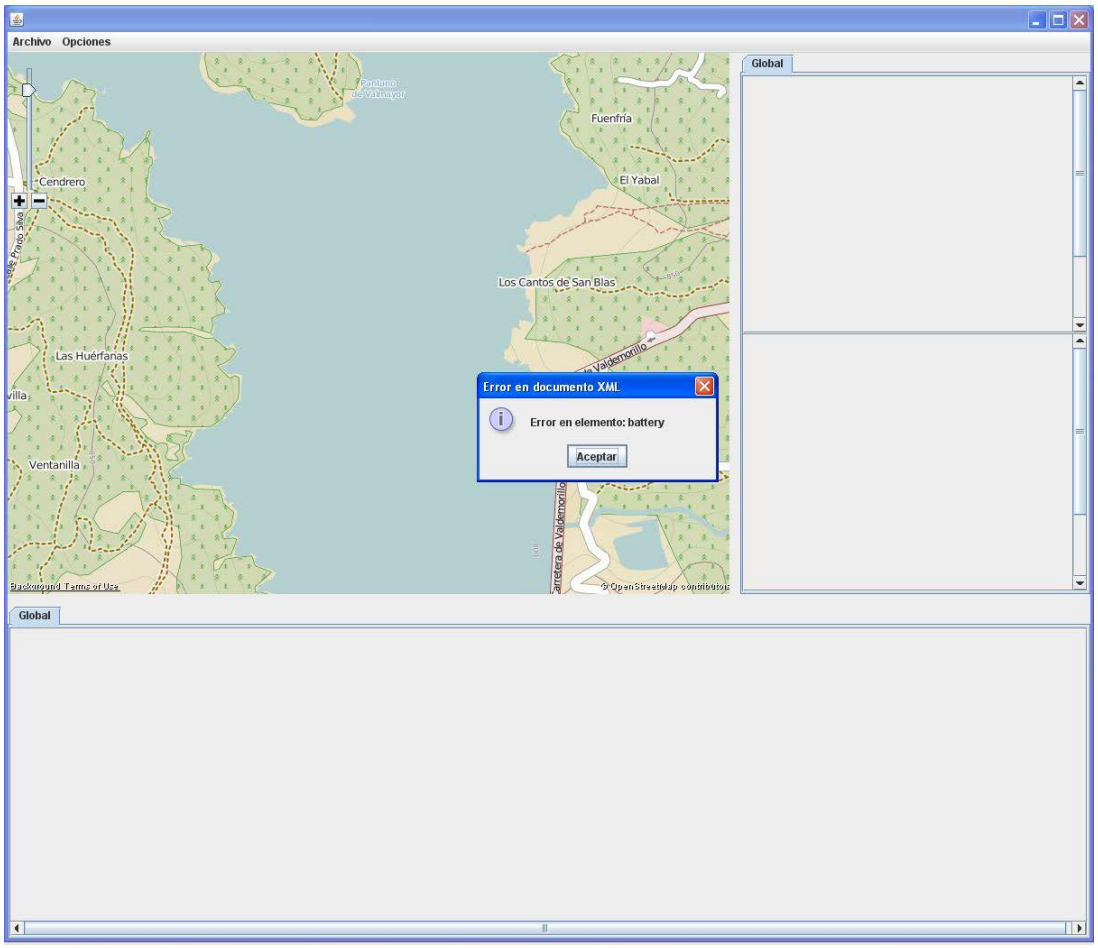

Figura 23: Ventana de Error en Documento XML

4.3.3 Manipulación y Asignación de Datos

Una vez comprobada la corrección de la DTD, se realiza una segunda lectura. Se planteó la posibilidad de realizar un almacenamiento simultáneo a la comprobación previa pero se desestimó ya que no provocaría una reducción notable en el tiempo de carga de los componentes del centro de control debido a la rapidez de lectura. Además ralentizaría esta primera comprobación y los elementos almacenados serían inútiles en caso de DTD incorrecta en esta segunda lectura. En la que se mapean los identificadores de variable de cada elemento del XML a un indicador o controlador en el que se irán cargando los valores que se reciban del software de control del vehículo autónomo.

Esta operación de mapeo se realiza mediante un método que se encarga de identificar el tipo de elemento, asignarle un objeto Java que representa a un indicador o controlador, almacenar el identificador y tipo de datos que guardará y establecer un sistema de punteros a todas las imágenes visibles de ese objeto para actualizarlo cada vez que se reciba un nuevo dato.

Esta actualización se realiza de forma considerablemente eficiente, ya que busca sólo los objetos ligados al identificador que se recibe junto al valor de la variable y los actualiza automáticamente gracias al interfaz **Updatable** que deben implementar todos los elementos visuales que han de ser actualizables.

CAPÍTULO 5: RESULTADOS EXPERIMENTALES

En este capítulo analizaremos los resultados obtenidos durante los ensayos realizados con el Centro de Control Versátil durante las sucesivas fases de desarrollo experimentadas así como algunos ajustes realizados a ciertos parámetros y código para mejorar el rendimiento.

5.1 Equipo y Software

Para la ejecución de los ensayos sobre el Centro de Control Versátil se ha utilizado una máquina virtual de Windows XP a la que se le ha asignado un núcleo Intel Core i7 4500-U corriendo a una velocidad máxima de 1.8Ghz, 2Gb de memoria RAM tipo DDR3, una capacidad de 20Gb de disco duro tipo SSD y una tarjeta gráfica integrada Intel HD Graphics 4400.

Tanto para las primeras etapas de los ensayos en los que se requería simulación de un PLC (Programmable Logic Controller) y del código de los vehículos autónomos como para los ensayos reales con vehículos autónomos (en este caso dos USV's simultáneos) se ha utilizado TwinCAT Service System v. 2.11.0 (Beckoff) [26], un software que presenta una gran compatibilidad y que convierte un PC en un controlador en tiempo real para sistemas multi-PLC. También cuenta con Hardware compatible, utilizado en los USV's antes mencionados, además de entorno de programación, posibilidad de conexión vía bus y comunicaciones con otros programas o interfaces de usuarios mediante los estándares Microsoft (p. ej. OPC, OCX, DLL).

Es importante destacar que el Hardware asignado a la máquina virtual es muy limitado si se compara con los equipos disponibles actualmente ya que incluso los PC's de características más asequibles o incluso muchos tablet PC's, normalmente menos potentes, cuentan con más de un núcleo y en la mayoría de las ocasiones con mayor cantidad de memoria RAM, disco duro y una tarjeta gráfica más potente. Esto nos puede dar idea de la poca cantidad de recursos necesarios para ejecutar el software del Centro de Control Versátil (tanto interfaz de usuario como Servidor y Clientes) ya que incluso siendo ejecutado en un entorno tan limitado y junto a otros programas que pueden consumir más recursos (p. ej. TwinCAT y el software de control) que normalmente estarían alojados en otro PC o PLC, el programa desarrolla su actividad con gran fluidez, sin observar retardos o pérdida de datos debido a falta de recursos.

Respecto al Software, como ya se ha comentado anteriormente, el lenguaje elegido para el desarrollo del Centro de Control Versátil es Java ya que es un lenguaje muy extendido, con un extenso API y multitud de proyectos de código abierto que nos ayudarán en el desarrollo del proyecto.

5.2 Ensayos Previos

El primer paso a la hora de comenzar con el desarrollo del Centro de Control Versátil fue elegir las herramientas y API's de código abierto apropiadas con las funcionalidades necesarias para lograr los objetivos propuestos.

Una vez realizadas estas tareas, se comenzó el desarrollo de los tres bloques de componentes que una vez ensamblados forman el Centro de Control Versátil.

En primer lugar se diseñó el interfaz gráfico modular. Se desarrollaron los componentes visuales uno a uno hasta que, una vez creado el último, pudieron unirse para formar el visualizador del Centro de Control Versátil. En esta parte del desarrollo se realizaron pruebas de integración con enfoque ascendente, es decir, se desarrolló y probó cada componente por separado para, posteriormente, unirlos dentro de elementos contenedores. Sobre cada uno de estos grupos se realizaron pruebas y, de esta forma, se fueron uniendo módulos cada vez mayores hasta formar la GUI del Centro de Control Versátil. Además del correcto funcionamiento y visualización de los componentes, fue necesario evaluar la ergonomía de los componentes obtenidos, ya que era éste uno de los objetivos propuestos en el proyecto. Solventando poco a poco los pequeños problemas surgidos a raíz de estas pruebas, se logró finalmente un resultado satisfactorio.

Otro de los ensayos más importantes tuvo lugar al desarrollar el segundo bloque de componentes, el configurador XML. En primer lugar se diseñó un esquema definido para el archivo de configuración XML, en base al cual se desarrolló un programa capaz de leer un documento XML y de tratar los componentes entrantes para formar el interfaz gráfico según las instrucciones del usuario. Además, para asegurar la correcta formación del archivo XML, se desarrolló un método que asegurase la correcta formación de cada uno de los elementos (incluidos sus atributos) del XML. Esto fue necesario porque JDOM no cuenta con un comprobador de corrección para el DTD. Para comprobar el funcionamiento de este módulo de configuración, se procedió a la realización de varios documentos XML incorrectos, en los que se alternaban múltiples tipos de errores, y varios documentos XML correctos para comprobar si se cargaban correctamente. Una vez resuelto este apartado satisfactoriamente, se procedió a integrar el interfaz gráfico y el configurador, de forma que se pudo comenzar a realizar pruebas de carga de interfaces gráficos modulares desde archivos XML, realizando sucesivas iteraciones de depuración hasta lograr el resultado deseado.

El tercer tipo de pruebas previas al ensamblado final de los tres bloques del proyecto, se diseñó con el objetivo de realizar una primera aproximación a lo que podría ser la visualización en tiempo real de un vehículo en OpenStreetMaps.

Para ello se comenzó el desarrollo del sistema de comunicaciones, un esquema sencillo cliente-servidor capaz de enviar y tratar datos para mostrarlos en el mapa.

Una vez logrado este objetivo, se comenzó la modificación y creación de las clases necesarias de OpenStreetMaps para cambiar su comportamiento añadiendo un método que actualizase la posición de un MapMarker en el mapa cuando recibiese nuevas coordenadas y otro que sustituyese el dibujo inicial, un círculo, por una imagen a escala a nuestra elección (durante la fase de pruebas un barco que representa al USV) y lo ubicase en la dirección correcta en cada caso.

Para realizar este primer ensayo, se preparó un código experimental que extraía, de un documento tipo TXT, las coordenadas de longitud, latitud y orientación de la trayectoria real de un USV y las introducía en el mapa a una velocidad superior a la necesaria en los experimentos reales (en condiciones normales se solicita una actualización de datos cada segundo) para comprobar su correcto funcionamiento en condiciones peores a las habituales. Se probaron varios intervalos distintos de actualización (cada 1 milisegundo y cada 250 milisegundos) sin observar fallos en el refresco de la pantalla.

El resultado fue satisfactorio ya que, el envío de coordenadas cliente-servidor se realizaba correctamente y además el nuevo MapMarker creado se movía por el mapa de forma fluida, sin presentar problemas como paradas inesperadas del programa o excepciones en el código, por lo que la decisión tomada fue seguir adelante con el diseño del proyecto contando con el API proporcionado por OpenStreetMaps, JOSM [27] y JMapViewer [28] [19] (más detalles en Apéndice OpenStreetMaps).

Adicionalmente, gracias a que las coordenadas introducidas pertenecían a la trayectoria real de un USV, se comprobó que la imagen sobre el mapa realizaba la ruta correctamente.

5.3 Ensayos con PLC Simulado en Entorno Local

Una vez concluidos los ensayos previos y desarrollada la mayor parte del software del Centro de Control Versátil (interfaz de usuario con mapas e indicadores, configurador, cliente y servidor), se consideró oportuno comenzar los ensayos con un barco simulado y controlado en un entorno local mediante los PLC's de TwinCAT. De hecho para aproximar el comportamiento al caso real se utilizó como PLC de control y simulación de un barco el mismo que utilizan los miembros del grupo de Ingeniería de Sistemas Control Automático y Robótica (ISCAR) para controlar barcos reales y simulados dentro de los proyectos: ***Sistema de vigilancia, búsqueda y rescate en el mar mediante colaboración de vehículos autónomos marinos y aéreos. (CICYT DPI2009-14552-C02-01)*** y ***Sistema Autónomo para la Localización y Actuación ante Contaminantes en el Mar (CICYT DPI2013-4665-C2-01)***.

Este tipo de ensayos suponen una importante aproximación al comportamiento real del centro de control conectando a uno o varios vehículos emisores de datos, ya que se trata de una simulación que reproduce unas condiciones muy similares a las reales, a excepción de los posibles retardos producidos por las comunicaciones, ya que el comportamiento del vehículo se simula con TwinCAT en entorno local y todas las comunicaciones se realizan en localhost.

Para la configuración de estas pruebas, el entorno elegido es la máquina virtual descrita en la sección 5.1. Determinado el equipo, el primer paso a realizar es ejecutar el código de simulación y control del PLC de los USV's con TwinCAT. El segundo paso es ejecutar el software controlador que enviará las instrucciones a los vehículos para arranque, parada, aceleración, deceleración, cambios de dirección, y maniobras preprogramadas que hacen que el PLC calcule las señales de

control que hay que aplicar a la hélice y timón del barco y simule su desplazamiento y los valores recogidos por los sensores acoplados al mismo.

Las primeras pruebas se realizaron con una configuración sencilla, consistente en un único USV, con un cliente que realizaba una lectura de variables al PLC y su envío al servidor cada 250 milisegundos. El resultado fue satisfactorio, ya que tanto el movimiento de la imagen del vehículo por el mapa como la actualización de indicadores (velocímetro y otros parámetros numéricos) se realizaban de forma fluida y sin cortes. En la Figura 24 podemos ver un ejemplo de prueba con simulación local, en la que el Cliente recoge los datos y los envía al centro de control que se muestra a la izquierda, donde se monitorizará el USV (movimiento en el mapa, velocidad, longitud, latitud, etc.). A la derecha se sitúa el panel de control del USV proporcionado por el grupo ISCAR mediante el cual podemos enviar ordenes al USV. En la parte superior derecha podemos ver una gráfica del movimiento realizado por el USV desde que se puso en marcha el motor.

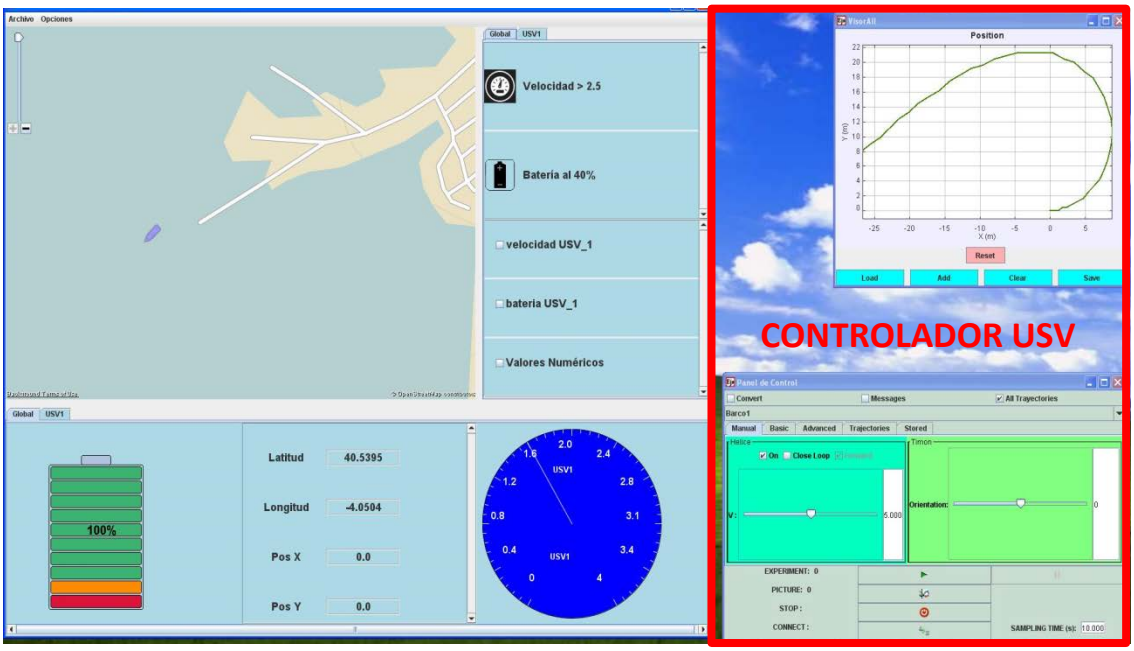

Figura 24: Ejemplo Prueba con Simulación Local

Posteriormente se configuró una prueba con dos vehículos de similares características a los del primer experimento, con un cliente para cada vehículo (y por tanto un canal y un hilo de actualización de variables dedicado para cada uno de ellos). Esta configuración comenzó a dar problemas debido a la modificación concurrente de ciertas estructuras que forman parte de clases de OpenStreetMaps (JMapViewer) [28] [19]. Estos problemas impedían una correcta visualización del interfaz de usuario ya que causaban gran número de excepciones (recordemos que se reciben 4 estructuras de coordenadas por segundo por cada vehículo) que provocaban una parada no deseada en el programa.

Para solventar este problema fue necesario acudir al código fuente de las clases antes mencionadas y realizar modificaciones sobre las estructuras y métodos accesores y mutadores de las mismas para adecuarlas a las necesidades del Centro de Control Versátil, que requería utilizar estructuras y métodos específicamente diseñados para soportar programación concurrente. Una vez realizado este

trabajo, se consiguió un programa lo suficientemente robusto para soportar la actualización simultánea de varios hilos sin ningún tipo de corte o excepción en el código.

De esta forma se consiguió llevar a cabo una serie de pruebas con la simulación de dos USV's simultáneos dando el software un rendimiento muy similar a la prueba con un solo USV y, por tanto, con unos resultados muy positivos.

5.4 Ensayo Real en Exterior con Múltiples Vehículos

Una vez probado el comportamiento del Centro de Control Versátil con una simulación en entorno local y cumplidos los objetivos marcados en cuanto a rendimiento (desarrollo fluido de la aplicación, sin errores, sin excepciones, sin pérdida de datos, etc.) se consideró oportuno realizar el experimento con vehículos reales.

Los USV's disponibles para el experimento serán los utilizados en los proyectos mencionados en la sección anterior.

La configuración del experimento es similar a la utilizada en experimentos anteriores. Se Ejecutará el interfaz de usuario del Centro de Control Versátil sobre la máquina virtual, junto con un servidor y dos clientes. Cada uno de estos clientes debe conectarse mediante las primitivas disponibles al PLC de su USV correspondiente. En este caso, se realiza la conexión con los vehículos por medio de una red wi-fi dedicada con un alcance de 5 Km, formada por dos antenas Ubiquiti modelo PicoStation M [29] montadas en los USV's y una antena Ubiquiti NanoStation M [30] en la zona del centro de control.

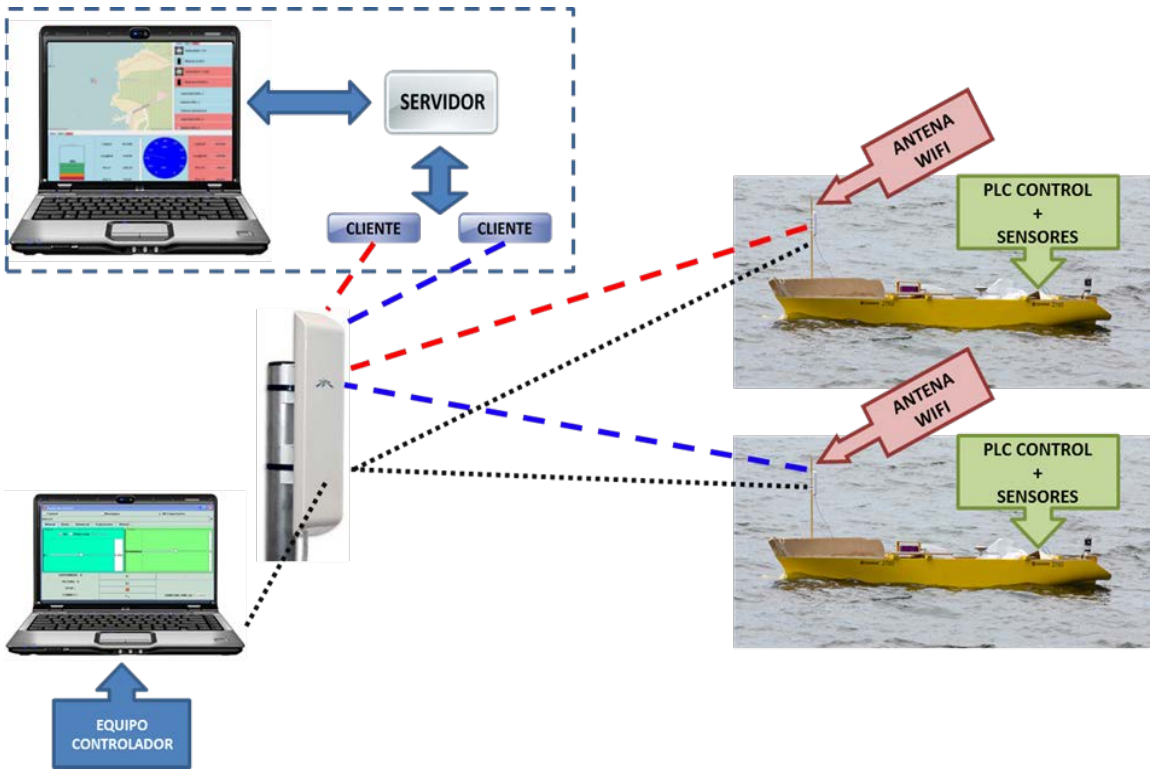

Figura 25: Esquema de las Pruebas Reales

El Equipo y los barcos se conectarán a esa red wi-fi de la forma ilustrada en la Figura 25, en la que también aparece (en la parte inferior izquierda) un equipo adicional sobre el que se ejecuta el software encargado de enviar las ordenes de control de alto nivel a los PLC's de los barcos.

Al arrancar cada cliente, éste realizará la petición de apertura de canal al servidor que la aceptará al comprobar la corrección de los datos enviados y conectará con su PLC correspondiente. Una vez realizadas las conexiones, el cliente realizará lecturas periódicas de datos de telemetría y los enviará al servidor que se encargará de su tratamiento y actualización en el interfaz de usuario. En este caso, se realizan las lecturas y reenvío de datos una vez por segundo (4 veces más lento que en experimentos anteriores), ya que se ha comprobado que es más que suficiente para lograr una actualización fluida tanto en el mapa como en el resto de indicadores del Centro de Control Versátil y que de esta forma se evita una sobrecarga innecesaria en los PLC's de los USV's.

En la captura del Centro de Control Versátil de la Figura 26, podemos observar un interfaz diseñado para monitorización de USV's. Los colores asignados automáticamente tanto para indicadores como para vehículos mostrados sobre el mapa (azul para USV1 y rojo para USV2), permiten distinguir rápidamente a cuál de los vehículos pertenece el indicador. También puede observarse un marcador amarillo, que representa una boya, introducido por el operador.

Figura 26:Captura CCV, Prueba Real (Pestaña Global) + Ventana Emergente

Los resultados obtenidos en este experimento han sido muy prometedores, ya que a pesar de ser la primera vez que se realiza con parámetros 100% reales (retardos en comunicaciones con USV's a una distancia de decenas e incluso cientos de metros, peticiones del cliente a PLC's de los vehículos en intervalos relativamente lentos respecto a los anteriores, retardos en la reacción de los USV's sobre el terreno, posibles interferencias y pérdidas de cobertura habituales en localizaciones exteriores sobre superficies acuáticas, etc.) el Centro de Control Versátil ha ofrecido una fluidez sobresaliente, además de una robustez y estabilidad notables durante las cuatro horas de ejecución continua que duraron los experimentos.

CAPÍTULO 6: CONCLUSIONES Y TRABAJO FUTURO

En este capítulo se analizarán las conclusiones posteriores al diseño, desarrollo y experimentación realizados sobre el Centro de Control Versátil además de los posibles campos de investigación para trabajo futuro.

6.1 Conclusiones

Al comienzo del proyecto, se realizó una primera fase de estudio de mercado, en la que hemos podido comprobar que son muy pocos los centros de control que ofrecen la posibilidad de monitorizar y controlar vehículos autónomos de naturaleza heterogénea y, los que lo hacen, suelen ser de tipo comercial o militar y por lo tanto de acceso muy restringido y por supuesto no reconfigurables. Por estas razones, creemos que la posibilidad de diseñar un Centro de Control Versátil que permita la inclusión simultánea de múltiples vehículos heterogéneos para su monitorización y control en tiempo real, es un campo de estudio de gran interés.

Durante el desarrollo del proyecto, se han logrado cumplir los objetivos marcados en las especificaciones del mismo e incluso mejorar algunos y añadir funcionalidades adicionales que fueron surgiendo durante la ejecución de las pruebas y ensayos realizados.

En primer lugar, se ha conseguido modificar OpenStreetMaps [5] para que mediante la modificación sobre el código fuente de algunas de sus clases y la agregación de otras nuevas sea capaz de mostrar varios vehículos de forma simultánea y en tiempo real sobre sus mapas, ya sean estos descargados previamente (para que se puedan usar en modo fuera de línea en caso de no contar con conexión a internet) o para su versión en línea. Además se ha habilitado el marcador que representa al vehículo para poder mostrar una imagen representativa a escala que muestre la dirección que sigue el vehículo en base a la orientación recibida.

También se ha logrado diseñar un interfaz visual de usuario que, mediante una serie de módulos independientes (pueden ser cargados o no dependiendo de la configuración elegida) monitoriza el estado del vehículo de forma fiable y precisa.

El interfaz de usuario es totalmente reconfigurable gracias al diseño del sistema de configuración. Este sistema modificará de forma transparente algunos parámetros referentes a las comunicaciones y a la cantidad y tipo de vehículos incluidos. Sin embargo, la característica más importante del mismo es que permite al usuario, mediante la especificación de un fichero XML, indicarle al interfaz los componentes necesarios, las variables del vehículo asociadas a ellos y su disposición en la

pantalla. Estas modificaciones se realizarán atendiendo a las necesidades derivadas de la misión a realizar, el vehículo al que represente y las preferencias del operador, mejorando la ergonomía y facilitando por tanto el desarrollo de su trabajo. El programa de lectura del XML se asegurará de la corrección del archivo de configuración gracias al sistema de verificación implementado, que avisará al usuario en caso de error mediante un mensaje en el que indicará el elemento en el que se ha detectado el fallo.

Gracias al esquema de comunicaciones cliente-servidor que se puede distribuir en múltiples configuraciones, se ha logrado obtener una gran flexibilidad a la hora de incorporar al sistema centros de control adicionales o incluso visores a los que un servidor redirige los datos recibidos de uno o varios vehículos. Todos estos equipos recibirán los datos en tiempo real.

Otro de los requisitos desarrollado en el Centro de Control Versátil consiste en la definición de alarmas en el archivo de configuración XML. Estas alarmas se vinculan a ciertas variables monitorizadas y avisan al operador si se alcanzan determinados valores de las mismas. De esta forma el operador puede centrar su atención en otros factores del desarrollo de la misión y reaccionar a tiempo ante cualquier situación crítica derivada del alcance de los valores antes mencionados (p. ej. batería baja, velocidad demasiado alta, etc.).

Se ha incluido también, la posibilidad de integrar en el interfaz de usuario los controladores ya existentes para los vehículos que se monitoricen, de modo que se permite interactuar con ellos y transformar el software en un centro de control y no una mera estación de monitorización.

Además de los objetivos principales establecidos al inicio del proyecto, a lo largo del desarrollo y en las sesiones de pruebas realizadas con el Centro de Control Versátil, se han introducido algunas mejoras y funcionalidades adicionales que se han considerado útiles o necesarias para el correcto desarrollo de las misiones.

En el apartado de interfaz de usuario se han incluido opciones como la posibilidad de, a petición del operador, mostrar y ocultar en el mapa la trayectoria que ha seguido un vehículo durante la misión. La ruta también podrá ser reiniciada (borrada y comenzada desde cero) en cualquier momento.

También se podrá introducir hitos (marcadores) en el mapa en tiempo de ejecución por medio de un interfaz en el que se introducirá las coordenadas (longitud y latitud) además de un identificador si es que se desea asignarle uno. De esta forma se hace posible señalizar ciertos puntos importantes para el desarrollo de una misión (p. ej. punto de inicio, objetivo, obstáculo, etc.).

Respecto a mejoras en el apartado de ergonomía, se ha añadido la posibilidad de extraer algunas partes de la ventana principal del interfaz (un solo indicador o incluso una pestaña vertical u horizontal de vehículo completa), creando un duplicado que se muestre en una ventana emergente que permita ampliar el espacio visible simultáneo del Centro de Control versátil.

Otra de las mejoras incluidas es un sistema de recopilación de datos en el centro de control durante la ejecución de misiones. Se ha habilitado una estructura que guarda temporalmente todos los datos recibidos desde los vehículos incluidos en el sistema y una opción en el menú Archivo para el que el operador pueda guardar los datos en cualquier momento que considere oportuno.

Consideramos, además, cumplidos los criterios establecidos en cuanto a robustez y fiabilidad ya que, por una parte, se ha establecido un sistema de verificación en la corrección de las especificaciones del archivo de configuración XML, lo que garantiza que el programa no sufrirá errores debido a mala asignación de variables, utilización errónea de alguno de los componentes o incluso incompatibilidad en la ubicación de los mismos en la pantalla. Por otra parte, a la hora de realizar las conexiones de comunicación, se ha establecido un protocolo de verificación que, a partir de las características del XML proporciona al Centro de Control Versátil la información necesaria para confirmar que las conexiones aceptadas son legítimas y garantizando la consistencia de datos en ambas partes, ahorrando posibles intrusiones y errores de funcionamiento.

También consideramos cumplidos los criterios de Versatilidad, ya que gracias al archivo de configuración XML y el interfaz gráfico modular, el Centro de Control Versátil permite la inclusión de uno o varios vehículos de naturaleza heterogénea además de permitir una enorme flexibilidad en cuanto a plataformas, lenguajes de programación y hardware utilizado.

Por lo tanto, y dado que además de cumplir los objetivos marcados, se han agregado funcionalidades adicionales, podemos concluir que el desarrollo del prototipo ha sido satisfactorio y que hemos logrado sentar las bases para un Centro de Control Versátil aún más avanzado, dejando abierto un interesante campo de investigación.

6.2 Trabajo Futuro

El desarrollo del software puede contemplarse como una espiral de mejora continua en la que se definen los requisitos del programa, se implementa el código, se realizan pruebas en base a las cuales se localizan carencias, errores y posibles mejoras para retornar al principio de la misma una y otra vez hasta conseguir un producto final que se ajuste lo más posible a la idea inicial que se pretendía.

Uno de los requisitos iniciales de este proyecto era crear un prototipo de centro de control que permitiese mostrar la utilidad del diseño desde el primer momento así como la posibilidad de añadir ampliaciones y mejoras futuras.

Ya durante el desarrollo y ensayos de este prototipo han surgido multitud de mejoras que se han ido incorporando tanto en el apartado gráfico (mejoras visibles y utilizables por el usuario final como las comentadas posibilidades de ventanas emergentes o extracción de pestañas) como en el apartado de optimización de código en el que modificando ciertos parámetros se puede mejorar el rendimiento del software.

Por lo tanto, el primer paso lógico a seguir en un futuro próximo es continuar los ensayos del software con vehículos autónomos reales en misiones sobre el terreno, ya que resultan muy beneficiosos a la hora de encontrar posibles fallos y localizar puntos críticos en los que sea necesario añadir funcionalidades o mejorar las ya existentes. Además, aumentando el número de vehículos y la heterogeneidad de los mismos, las sesiones de ensayo serían mucho más provechosas dado que las posibilidades de añadir modificaciones o mejoras aumentan exponencialmente con cada vehículo.

Respecto al interfaz gráfico de usuario (GUI) hay varios puntos en los que se puede seguir trabajando para mejorar el Centro de Control Versátil. Una de las primeras mejoras importantes que surgen de forma inmediata es continuar con el desarrollo de nuevos indicadores visuales estándar y personalizables. De esta forma se logrará cubrir un mayor rango de necesidades que puedan derivarse de la inclusión en el sistema de nuevos vehículos aún no contemplados. Otra posible mejora del apartado visual sobre la que se puede trabajar es la inclusión de nuevos tipos de gráficas (además de mejorar las ya incluidas) que ayuden a mejorar el análisis y comprensión de los datos que se están recibiendo de los vehículos o de los que ya han sido recopilados. Para esto contamos con una API de JFreeChart [22], que permite mostrar los datos en numerosos tipos de gráficos predefinidos.

Otro punto del interfaz visual en el que nos gustaría trabajar es la mejora de la precisión a la hora de mostrar vehículos, rutas y marcadores sobre el mapa ya que, aun pareciendo más que aceptable la resolución actual, consideramos interesante trabajar más en la escala de dichos componentes para contar con una referencia visual lo más precisa posible.

También referente a mejoras sobre visualización en el mapa, resultaría muy interesante introducir marcadores y límites desde un archivo XML. De este modo se mejoraría la posibilidad de introducir marcadores en tiempo real si éstos son muy numerosos o quieren utilizarse para delimitar ciertas zonas del mapa (p. ej. posibles zonas de colisión, objetivos, zonas de inicio o final, hitos que alcanzar durante una ruta). Una vez introducidos estos límites, también sería altamente recomendable introducir desde las opciones de ese XML una serie de alarmas que se activasen al alcanzar estos límites o coordenadas concretas para señalizar peligro, final de misión o alcance de un objetivo.

Si hablamos de las comunicaciones, sería interesante seguir mejorando los protocolos actuales tanto en seguridad como en optimización y estudiar formas de introducir en el software compatibilidad para algunos nuevos protocolos que puedan resultar útiles. De esta forma algunos vehículos que se comuniquen mediante protocolos que por alguna razón no fuesen compatibles, podrían incluirse en el sistema. Sería también muy provechoso estudiar la posibilidad de ampliar la red de comunicaciones con nuevo hardware como sistemas infrarrojos para comunicar vehículos entre sí a cortas distancias de forma que puedan intercambiar datos durante el desarrollo de misiones, señales de radio o alguna de las otras posibilidades que se han mencionado durante los estudios previos al desarrollo del proyecto.

Respeto al sistema de configuración, las mejoras y ampliaciones vendrán dadas por la inclusión de nuevos componentes necesarios para el Centro de Control Versátil, ya sea nuevos módulos gráficos o parámetros de configuración. Estas ampliaciones, gracias a la potencia del esquema creado, resultan casi inmediatas añadiendo sólo unas pocas líneas de código y un nuevo elemento al esquema de definición del archivo XML.

Otra posible mejora interesante sería la posibilidad de incluir un interfaz gráfico sencillo que permitiese añadir elementos al XML desde una ventana preparada para ello y en la que el usuario se despreocupase de la sintaxis (sistema de etiquetas de XML), para introducir sólo los valores de cada campo del elemento,

quedando estos verificados en el momento de aceptar la inclusión del mismo o una herramienta tipo drag & drop (arrastrar y soltar) que permita seleccionar elementos que se añadan al XML por medio de un interfaz gráfico.

Una importante mejora en lo que concierne a los elementos de control de vehículos, sería el estudio de una mayor integración del software de control y el Centro de Control Versátil. De esta forma, integrando los controles directamente en el interfaz de usuario y creando un interfaz que permita acceder a las primitivas que controlan los vehículos, podría lograrse una mayor ergonomía. Además, esta posibilidad permitiría incluir mejoras muy notables como la posibilidad de marcar con el ratón un punto en el mapa y ordenar al vehículo que se dirija a él, o restringir una zona colocando marcadores y automatizando una parada en base a su proximidad. También es importante destacar que es necesario realizar la implementación en el Centro de Control Versátil de un sistema de prioridad para evitar posibles conflictos cuando haya más de un centro de control actuando sobre los vehículos que se encuentran desarrollando una posición. Para solucionar este problema se ha pensado la inclusión de un sistema de seguridad que habilite sólo a uno de los centros de control disponibles para realizar labores de control. Al seleccionar el "modo control", el centro de control que lo activa enviará una señal al resto que inhabilitará su panel de control (previo aviso al operador) para que las posibles órdenes enviadas desde varias ubicaciones no entren en conflicto. También podría estudiarse un sistema de prioridades en las que un controlador pudiese enviar órdenes y el cliente ligado a cada vehículo se encargase de desecharlas en caso de que haya una orden de mayor prioridad actuando en el momento. Consideramos este sistema de prioridades uno de los puntos clave en un centro de control de este tipo, por lo que es uno de los que requerirán una mayor carga de trabajo.

En cuanto a la recolección y posterior tratamiento de datos, una mejora interesante que ya se ha planteado es la posibilidad de seleccionar los datos que desean guardarse y el formato en que se guardan los mismos. En la actualidad el sistema recopila y ofrece la posibilidad de guardar la totalidad de los datos en un fichero .txt, que posteriormente se distinguirán por el identificador del vehículo al que pertenecen. La mejora propuesta, sería desplegar una ventana en la que se mostrasen todos los vehículos junto a las variables que se reciben de cada uno para poder seleccionar exactamente cuáles de ellos se quieren guardar en cada momento.

También podría resultar provechoso realizar una conexión con algún tipo de base de datos (ya sea local o remota vía internet) y la automatización del almacenamiento de datos en ella por períodos constantes de tiempo. De esta forma los resultados de los experimentos quedarían guardados en un almacenamiento persistente, ya clasificados y relacionados para una mayor comodidad en su consulta.

Por último, la posibilidad de configurar diferentes entornos gráficos de usuario nos permitiría estudiar qué esquemas gráficos son más adecuados dependiendo del tipo de vehículo monitorizado, número de vehículos, alarmas definidas, etc. De este modo podrían proponerse esquemas de configuración que faciliten el trabajo

a los operadores que lo vayan a utilizar dependiendo del tipo de misión que se vaya a llevar a cabo.

Mediante los avances propuestos y la continuación del proceso de mejora continua, se considera que el Centro de Control Versátil podría seguir creciendo hasta convertirse en una solución estándar adaptable a un número de situaciones prácticamente ilimitado y, por tanto, una buena solución para el problema de monitorización y control de múltiples vehículos de naturaleza heterogénea en tiempo real.

BIBLIOGRAFÍA

[1] «barnardmicrosystems,» [En línea]. Available: http://www.barnardmicrosystems.com/L4E_ground_control.htm . [Último acceso: 10 Mayo 2014].

[2] «ga-asi.com,» [En línea]. Available: http://www.ga-asi.com/products/ground_control/mobile.php. [Último acceso: 10 Julio 2014].

[3] «QGroundControl,» [En línea]. Available: http://qgroundcontrol.org/. [Último acceso: 27 mayo 2014].

[4] Pérez, I. Maza y F. Caballero, «Ground Control Station for a Multi-UAV Surveillance,» *Intelligent & Robotic Systems,* vol. 69, nº 1-4, pp. 119-130, 2013.

[5] «OpenStreetMaps,» [En línea]. Available: http://www.openstreetmap.org. [Último acceso: 15 Marzo 2014].

[6] «ign,» [En línea]. Available: http://www.ign.es . [Último acceso: 1 Mayo 2014].

[7] «UAS-Europe (3),» [En línea]. Available: http://www.uas-europe.se/index.php/products/skyview-ground-control-station-software. [Último acceso: 20 Mayo 2014].

[8] «RockwellCollins.com (4),» [En línea]. https://www.rockwellcollins.com/Products_and_Systems/Communications_and_Networks/Video_Receivers.aspx . [Último acceso: 20 Mayo 2014].

[9] «UAS-Europe (2),» [En línea]. Available: http://www.uas-europe.se/index.php/products/portable-gcs-computer. [Último acceso: 20 Mayo 2014].

[10] «UCON(4),» [En línea]. Available: http://uconco.en.ec21.com/Portable_Ground_Control_System_PGCS--865587_865591.html . [Último acceso: 20 Mayo 2014].

[11] «UAV Factory (5),» [En línea]. Available: http://www.uavfactory.com/product/16. [Último acceso: 10 Julio 2014].

[12] «UCON(2),» [En línea]. Available: http://uconco.en.ec21.com/Ground_Control_System_GCS--865587_865588.html. [Último acceso: 20 Mayo 2014].

[13] «UCON(3),» [En línea]. Available: http://uconco.en.ec21.com/Tactical_Ground_Control_System_TGCS--

865587_865590.html. [Último acceso: 20 Mayo 2014].

[14] «SAAB Group,» [En línea]. Available: http://www.saabgroup.com/en/Air/Airborne-Solutions/Unmanned_Aerial_Systems/Skeldar_UAS_Control_Station/Technical_specifications/. [Último acceso: 20 Mayo 2014].

[15] «ga-asi.com (2),» [En línea]. Available: http://www.ga-asi.com/products/ground_control/fixed.php. [Último acceso: 10 Julio 2014].

[16] «ga-asi.com (3),» [En línea]. Available: http://www.ga-asi.com/products/ground_control/mobile.php. [Último acceso: 10 Julio 2014].

[17] «ArduPilot,» [En línea]. Available: http://ardupilot.com/. [Último acceso: 17 Marzo 2014].

[18] I. Maza, F. Caballero, R. Molina, N. Peña y A. Ollero, «Multimodal Interface Technologies for UAV Ground Control Stations. A Comparative Analysis,» *Journal of Intelligent and Robotic Systems,* vol. 57, nº 57, pp. 371-391, 2010.

[19] «JMapViewer - JavaDoc,» [En línea]. Available: http://josm.openstreetmap.de/doc/index.html?org/openstreetmap/gui/jmapviewer/JMapViewer.html. [Último acceso: 10 Mayo 2014].

[20] «JDom,» [En línea]. Available: http://www.jdom.org/. [Último acceso: 15 Marzo 2014].

[21] «Oracle - Paquete java.net,» [En línea]. Available: http://docs.oracle.com/javase/7/docs/api/java/net/package-summary.html. [Último acceso: 15 Mayo 2014].

[22] «JFreeChart,» [En línea]. Available: http://www.jfree.org/jfreechart/. [Último acceso: 4 Junio 2014].

[23] «World Wide Web Consortium,» [En línea]. Available: http://www.w3c.es/. [Último acceso: 10 Marzo 2014].

[24] «World Wide Web Consortium (XML),» [En línea]. Available: http://www.w3.org/standards/xml/. [Último acceso: 14 Junio 2014].

[25] «DesarrolloWeb.com (Manuales - XML),» [En línea]. Available: http://www.desarrolloweb.com/manuales/manual-introduccion-xml.html. [Último acceso: 10 Marzo 2014].

[26] «Beckoff TwinCat,» [En línea]. Available: http://beckhoff.com/english.asp?twincat/default.htm. [Último acceso: 6 Junio 2014].

[27] «JOSM,» [En línea]. Available: https://josm.openstreetmap.de/. [Último acceso: 10 Marzo 2014].

[28] «JMapViewer - wiki,» [En línea]. Available: http://wiki.openstreetmap.org/wiki/JMapViewer. [Último acceso: 3 Mayo 2014].

[29] «Ubiquiti - PicoStation M,» [En línea]. Available: http://www.ubnt.com/airmax#picostationm. [Último acceso: 12 Junio 2014].

[30] «Ubiquiti - NanoStation M,» [En línea]. Available: http://www.ubnt.com/airmax#nanostationm. [Último acceso: 12 Junio 2014].

[31] «OpenStreetMap Foundation,» [En línea]. Available: https://blog.openstreetmap.org/about/. [Último acceso: 15 Marzo 2014].

[32] «JMapViewer - OfflineTileSource,» [En línea]. Available: https://github.com/balloob/JMapViewer. [Último acceso: 10 Marzo 2014].

[33] «JTileDownloader,» [En línea]. Available: http://wiki.openstreetmap.org/wiki/JTileDownloader. [Último acceso: 10 Marzo 2014].

[34] «UCON,» [En línea]. Available: http://uconco.en.ec21.com/. [Último acceso: 20 Mayo 2014].

[35] «ga-asi.com Multi-Function GCS» [En línea]. Available: http://www.ga-asi.com/products/ground_control/multi_function_workstation.php [Último acceso: 10 Julio 2014].

[36] «UAS-Europe,» [En línea]. Available: http://www.uas-europe.se/. [Último acceso: 20 Mayo 2014].

[37] «wikipedia.org,» [En línea]. Available: http://es.wikipedia.org/wiki/Extensible_Markup_Language. [Último acceso: 12 Junio 2014].

APÉNDICES:

A continuación se incluyen dos apéndices sobre las dos API's más importantes utilizadas durante el desarrollo del proyecto, OpenStreetMaps (JMapViewer), utilizada para mostrar los mapas donde se visualizarán los vehículos y JDOM, utilizada para realizar la lectura del archivo de configuración XML.

Apéndice 1: OpenStreetMaps

Uno de los puntos clave, quizá el más crítico, a la hora de diseñar el interfaz gráfico de usuario fue elegir un mapa donde representar los vehículos ya que a la hora de diseñar un centro de control este tipo, resulta muy importante tener una referencia visual de la posición de los mismos.

Tras evaluar varios tipos de mapas, finalmente se eligió OpenStreetMaps [5], una iniciativa cuyo objetivo es crear y proporcionar datos geográficos libres, tales como callejeros y mapas de carreteras. La organización sin ánimo de lucro OpenStreetMap Foundation [31] se dedica a promover el crecimiento, desarrollo y distribución de datos geoespaciales libres, así como de proporcionar los medios para liberar datos geográficos libres para que cualquiera pueda usarlos y compartirlos.

Una de las herramientas de código abierto utilizadas para desarrollar el mapa del Centro de Control Versátil es JMapViewer [28], un componente java de código abierto que permite integrar OpenStreetMaps en nuestra aplicación, proporcionando todas los componentes necesarios para su utilización (p. ej. zoom, desplazamiento sobre el mapa, inclusión de marcadores estáticos, etc.).

Al ser JMapViewer una herramienta de código abierto, nos permite realizar las modificaciones y desarrollos necesarios para lograr un comportamiento adecuado para las necesidades de nuestra aplicación.

El primer elemento a modificar fue la fuente de carga de "tiles" (cuadrados con porciones de mapa), ya que por requisitos del proyecto (en ubicaciones exteriores no se cuenta con conexión a internet) se debía tener la posibilidad de mostrar los vehículos en un mapa en modo fuera de línea. Para realizar este cambio fue necesario descargar el código fuente de JMapViewer y modificar el código fuente añadiendo la clase "OfflineTileSource.java" [32] y recompilando para volver a generar la librería que posteriormente importaríamos en nuestro proyecto. De esta forma ya disponíamos de una herramienta con la que cargar a nuestro mapa los "tiles" de una fuente en nuestro propio equipo.

Para descargar los "tiles" deseados, se ha utilizado una herramienta llamaba JTileDownloader [33], que proporciona un sencillo interfaz desde el que seleccionar una zona del mapa y descargarla pulsando un botón.

El siguiente paso fue crear nuestro propio tipo de marcador, basado en MapMarkerDot, proporcionado por JMapViewer y que hemos llamado MapMarkerTriangle. Este marcador permite mostrar en el mapa una imagen elegida por nosotros que se introducirá en el constructor y la cargará con la orientación correcta sobre el mapa dependiendo de las coordenadas (Longitud, Latitud y Orientación) que reciba. En caso de no introducir la ruta de la imagen se mostrará un triángulo. Para que las imágenes sean válidas, debe haber un número concreto (hasta el momento se está utilizando 16 orientaciones diferentes para lograr la precisión necesaria del vehículo sobre el mapa) y estas deben nombrarse de una forma concreta que se indicará en la documentación de usuario (nombre de la coordenada a la que representa N – Norte, S – Sur, etc.).

Además se ha creado un sistema de actualización para dicho marcador que se gestiona mediante un método que será invocado al recibir los nuevos datos de posición y realizará el borrado de la imagen anterior y dibujado sobre el mapa de la posición actual. Una vez implementado el método, se realizaron una serie de pruebas para comprobar si era posible utilizarlo para realizar visualización de vehículos en tiempo real y se comprobó que gracias a este interfaz específico para manejar un flujo continuo de coordenadas y mostrarlo en el mapa, se logra crear una imagen en movimiento lo suficientemente precisa como para seguir el desarrollo de la trayectoria de un vehículo en tiempo real.

Durante el desarrollo de estas pruebas, comenzaron a surgir problemas debido a la inserción concurrente de coordenadas de dos fuentes distintas. Los errores (excepciones en el código) ocasionados, ocasionaban la parada de la aplicación y por tanto la parada de la imagen de los vehículos mostrados sobre el mapa.

Tras un estudio del código fuente de JMapViewer, se llegó a la conclusión de que no estaba diseñado para soportar programación concurrente, esto es, varios hilos accediendo de forma simultánea a las mismas estructuras de datos (recordemos que, como hemos explicado en secciones anteriores de esta memoria, nuestro esquema de comunicaciones cuenta con un canal dedicado abierto en un hilo independiente para que cada vehículo envíe sus datos de telemetría).

El problema se solucionó acudiendo una vez más al código fuente de JMapViewer y cambiando todas las estructuras a las que se debía acceder de forma concurrente, en su mayoría listas que guardaban marcadores a mostrar en el mapa. Originalmente estas estructuras eran de la clase **LinkedList**, una clase estándar java que no soporta acceso concurrente por estructuras tipo **Vector**, que implementan el interfaz "**Synchronized**" (Sincronizado) y que por lo tanto soportan programación concurrente.

Para una mayor seguridad, se sincronizó también los métodos accesores y mutadores de las estructuras modificadas añadiéndoles la cláusula "**synchronized**".

De esta forma, gracias a que OpenStreetMaps y JMapViewer son herramientas de código abierto, hemos podido utilizarlas para crear el Centro de Control Versátil mejorando algunas de sus clases y añadiendo otras de tal forma que su comportamiento se ajuste a los requisitos marcados en nuestro proyecto.

Apéndice 2: JDOM

Otro de los puntos clave en el desarrollo del Centro de Control Versátil es la inclusión de un archivo de configuración XML a partir del cual se modificarán características como parámetros de configuración, ubicación de componentes gráficos e inclusión de vehículos heterogéneos.

La herramienta elegida para realizar la lectura y tratamiento de la información de los archivos XML es JDOM [20], un proyecto que pretende proporcionar una solución simple y completa para acceso y manipulación de documentos XML en Java.

Esta herramienta de desarrollo permite extraer la información de los elementos (y atributos de los mismos) incluidos en un archivo XML mediante el uso de una serie de métodos sencillos e intuitivos.

Para la lectura de un archivo XML, debemos abrir el archivo utilizando la clase **SAXBuilder** de JDOM y una instancia del documento abierta desde una ubicación indicada del equipo en el que nos encontremos.

Una vez abierto el archivo, obtenemos el elemento raíz, **rootElement**, que es el elemento principal que contiene a todos los demás. En el caso de nuestro archivo de configuración, se ha decidido dar a la etiqueta del elemento raíz el nombre "config", quedando de la siguiente forma: **<config></config>**.

Una vez obtenido el elemento raíz debemos iterar recursivamente sobre éste hasta obtener todos los elementos que contiene y subelementos de éstos si es que existen.

Según se van obteniendo los elementos, se debe ir extrayendo la información que éstos contienen (texto entre las etiquetas, p. ej. <id>**identificador**</id>), además de la contenida en los atributos que puedan formar parte de ellos.

A continuación, en la Figura 27, podremos ver un sencillo ejemplo de código con los métodos utilizados por JDOM para obtener elementos, atributos y la información de los mismos. En el ejemplo se obtendrán elementos de un nivel inferior al elemento raíz (sus hijos directos), para realizar una lectura completa es necesario buscar recursivamente hijos en cada uno de los elementos obtenidos.

JDOM proporciona una potente herramienta para acceso y modificación de archivos XML, pero hasta el momento no se ha terminado de desarrollar ninguno de los proyectos en marcha que pretenden realizar un comprobador estándar para DTD (Document Type Definition) de XML's.

Por esta razón durante el desarrollo del Centro de Control Versátil nos hemos visto obligados a implementar nuestro propio comprobador de corrección, un método llamado **correctDTD**, que obtiene los elementos uno a uno y comprueba la corrección de su estructura de cada elemento mediante el método **correctElementStructure**, y de sus atributos mediante el método **correctElementAttributes**. Estos métodos reciben un Elemento y analizan su estructura (orden, contenido y cantidad de elementos y atributos), avisando del elemento incorrecto si es que hay alguno.

Una vez comprobado esto, se realiza la asignación de datos extraídos de los elementos y su asignación a objetos en del lenguaje seleccionado mediante el método **getElement.**

```
//Creamos objeto SAXBuilder de JDOM
SAXBuilder builder = new SAXBuilder();
//Creamos una instancia del documento ubicado en la ubicación "path" de nuestro equipo
xmlFile = new File(path);

    Document document=null;
    try {
        //Abrimos el documento con SAXBuilder y capturamos posibles excepciones
        document = (Document) builder.build(xmlFile);
    } catch (JDOMException ex1) {
        ex1.printStackTrace();
    } catch (IOException ex2) {
        ex2.printStackTrace();
    }
    //Se obtiene elemento raíz que contendrá a todos los demás
    Element rootNode = document.getRootElement();
    //Se obtiene elementos de nivel 1 (hijos de rootElement)
    List elementosNivel1= rootNode.getChildren();
    //Se itera sobre la lista para obtener hijos
    for (int i=0;i<elementosNivel1.size();i++){

        //Se obtiene elemento
        Element e=elementosNivel1.get(i);

        //Se trata información de sub-elementos
        //Ejemplo de obtención de la información en el nodo <identificador>USV_barco1</identificador>
        String identificador=e.getChild("identificador").getValue();

        //Se trata información de atributos
        //Ejemplo de obtención de la información de un atributo: type="USV"
        String tipo=vehicleType=e.getAttributeValue("type");

        /**
            Aquí debería realizarse la recursividad si se desea
            obtener todos los sub-elementos posibles
        **/
    }
```

Figura 27: Ejemplo de Código Java con Para Leer XML con JDOM